Math. Struct. in Comp. Science (2016), *vol.* 26, *pp.* 367–423. © Cambridge University Press 2014
doi:10.1017/S0960129514000218 First published online 10 November 2014

Linear logic propositions as session types

LUÍS CAIRES[†], FRANK PFENNING[‡] and BERNARDO TONINHO[†,‡]

[†]*Faculdade de Ciências e Tecnologia and CITI, Universidade Nova de Lisboa, Lisboa, Portugal*
Emails: `btoninho@gmail.com` and `luis.caires@fct.unl.pt`
[‡]*Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA*
Email: `fp@cs.cmu.edu`

Received 14 November 2012; revised 5 March 2013

Throughout the years, several typing disciplines for the π-calculus have been proposed. Arguably, the most widespread of these typing disciplines consists of session types. Session types describe the input/output behaviour of processes and traditionally provide strong guarantees about this behaviour (i.e. deadlock-freedom and fidelity). While these systems exploit a fundamental notion of linearity, the precise connection between linear logic and session types has not been well understood.
This paper proposes a type system for the π-calculus that corresponds to a standard sequent calculus presentation of intuitionistic linear logic, interpreting linear propositions as session types and thus providing a purely logical account of all key features and properties of session types. We show the deep correspondence between linear logic and session types by exhibiting a tight operational correspondence between cut-elimination steps and process reductions. We also discuss an alternative presentation of linear session types based on classical linear logic, and compare our development with other more traditional session type systems.

1. Introduction

Linear logic has been intensively explored in the analysis of π-calculus models for communicating and mobile system, given its essential ability to deal with resources, effects and noninterference. The fundamental way it provides for analysing notions of sharing versus uniqueness, captured by the exponential '!', seems to have been a source of inspiration for Milner when introducing replication in the π-calculus (Milner 1992). Following the early works of Abramsky (1993), several authors have exploited variants of π-calculi to express proof reductions (e.g. Bellin and Scott (1994)) or game semantics (e.g. Hyland and Ong (1995)) in systems of linear logic.

In the field of concurrency, many research directions have also drawn inspiration from linear logic for developing type-theoretic analyses of mobile processes, motivated by the works of Kobayashi *et al.* (1996); a similar influence is already noticeable in the first publications by Honda on session types (Honda 1993). Many expressive type disciplines for π-calculi in which linearity frequently plays a key role have been proposed since then (e.g. Giunti and Vasconcelos (2010), Honda *et al.* (1998), Kobayashi (1998) and Yoshida *et al.* (2007)). However, linearity has been usually employed in such systems in indirect ways, exploiting the fine grained type context management techniques it provides, or the assignment of usage multiplicities to channels (Kobayashi *et al.* 1996), rather than the deeper type-theoretic significance of linear logical operators.

In this paper, we present two type systems for the π-calculus that exactly correspond to the standard dyadic sequent calculus for intuitionistic linear logic (DILL) and for classical linear logic (CLL), respectively. The former was first introduced in Caires and Pfenning (2010) and studied in detail in this paper, the latter is introduced here.

The key to our correspondence is a new, perhaps surprising, interpretation of intuitionistic linear logic formulas as a form of session types (Honda 1993; Honda *et al.* 1998), in which the programming language is a session-typed π-calculus, and the type structure consists precisely of the connectives of intuitionistic linear logic, retaining their standard proof-theoretic interpretation. We thus introduce the first purely logical account, in the style of a Curry–Howard interpretation, of both shared and linear features of session types, as formulated for π-calculus-based session type systems such as Gay and Hole (2005).

In session-based concurrency, processes communicate through so-called session channels, connecting exactly two subsystems, and communication is disciplined by session protocols so that actions always occur in dual pairs: when one partner sends, the other receives; when one partner offers a selection, the other chooses; when a session terminates, no further interaction may occur. New sessions may be dynamically created by invocation of shared servers. Such a model exhibits concurrency in the sense that several sessions, not necessarily causally related, may be executing simultaneously, although races in unshared resources are forbidden; in fact this is the common situation in disciplined concurrent programming idioms. Mobility is also present, since both session and server names may be passed around (delegated) in communications. Session types have been introduced to discipline interactions in session-based concurrency, an important paradigm in communication-centric programming (Dezani-Ciancaglini and de' Liguoro 2010).

It turns out that the connectives of linear logic suffice to express all the essential features of finite session disciplines.

While in the linear λ-calculus types are assigned to terms (denoting functions and values), in our interpretation types are assigned to names (denoting communication channels) and describe their session protocol. The essence of our interpretation may already be found in the interpretation of the linear logic multiplicatives as behavioural prefix operators. Traditionally, an object of type $A \multimap B$ denotes a linear function that given an object of type A returns an object of type B (Girard and Lafont 1987). In our interpretation, an object of type $A \multimap B$ denotes a session x that first inputs a session channel of type A, and then behaves as B, where B specifies again an interactive behaviour, rather than a closed value. Linearity of $\multimap$ is essential, otherwise the behaviour of the input session after communication could not be ensured. An object of type $A \otimes B$ denotes a session that first sends a session channel of type A and afterwards behaves as B. But notice that objects of type $A \otimes B$ really consist of two objects: the sent session of type A and the continuation session, of type B. These two sessions are separate and non-interfering, as enforced by the canonical semantics of the linear multiplicative conjunction ($\otimes$). Our interpretation of $A \otimes B$ appears asymmetric, in the sense that, of course, a channel of type $A \otimes B$ is in general not typeable by $B \otimes A$. In fact, the symmetry captured by the proof of $A \otimes B \vdash B \otimes A$ is realized by an appropriately typed process that coerces any session of type $A \otimes B$ to a session of type $B \otimes A$. The other linear constructors are also given compatible interpretations, in particular, the $!A$ type is naturally interpreted as a type of a shared server for sessions of type A, and additive product and sum, to branch and choice session type operators. We thus obtain the first purely logical account of both shared and linear features of session types.

We briefly summarize the structure and contributions of the paper. After introducing our basic process model (Section 2), we describe a system of session types for the π-calculus that corresponds to the DILL (Section 3). The correspondence is bidirectional and tight, in the sense that (a) any π-calculus computation can be simulated by proof reductions on typing derivations (Theorem 4.1), thus establishing a strong form of subject reduction (Theorem 4.4) and (b) that any proof reduction or conversion corresponds either to a computation step or to a process equivalence on the π-calculus side (Theorems 4.2 and 4.3). An intrinsic consequence of the logical typing is a global progress property, that ensures the absence of deadlock for systems with an arbitrary number of open sessions (Theorem 4.5). In (Section 5) we propose a version of our linear session type structure based on CLL, and offer some preliminary comparison with the intuitionistic formulation. We close the technical part of the paper with some discussion comparing our session types based on linear logic with other traditional type systems for session types (Section 6). Finally, in Section 7 we comment on related work and present some closing remarks.

2. Process model

We briefly introduce the syntax and operational semantics of the process model: the synchronous π-calculus (Sangiorgi and Walker 2001) extended with (binary) guarded choice.

Definition 2.1 (processes). Given an infinite set Λ of *names* (x, y, z, u, v), the set of *processes* (P, Q, R) is defined by

$$\begin{array}{rcll}
P & ::= & \mathbf{0} & \text{(inaction)} \\
& \mid & P \mid Q & \text{(parallel composition)} \\
& \mid & (\boldsymbol{\nu} y)P & \text{(name restriction)} \\
& \mid & x\langle y\rangle.P & \text{(output)} \\
& \mid & x(y).P & \text{(input)} \\
& \mid & !x(y).P & \text{(replicated / shared input)} \\
& \mid & x.\mathtt{inl};P & \text{(left choice)} \\
& \mid & x.\mathtt{inr};P & \text{(right choice)} \\
& \mid & x.\mathtt{case}(P,Q) & \text{(case offer).}
\end{array}$$

The operators $\mathbf{0}$ (inaction), $P \mid Q$ (parallel composition), and $(\boldsymbol{\nu} y)P$ (name restriction) comprise the static fragment of any π-calculus. We then have $x\langle y\rangle.P$ (send y on x and proceed as P), $x(y).P$ (receive a name z on x and proceed as P with the input parameter y replaced by z) and $!x(y).P$ which denotes replicated (or persistent) input. The remaining three operators define a minimal labelled choice mechanism, comparable to the n-ary branching constructs found in standard session π-calculi. The restriction to guarded choice and replication is standard in the literature (see eg. Honda *et al.* (1998)).

For the sake of minimality and without loss of generality we restrict our model to binary choice. In restriction $(\boldsymbol{\nu} y)P$ and input $x(y).P$ the distinguished occurrence of the name y is binding, with scope the process P. For any process P, we denote the set of *free names* of P by $fn(P)$. A process is *closed* if it does not contain free occurrences of names. We identify processes up to consistent renaming of bound names, writing $\equiv_\alpha$ for this congruence. We write $P\{x/y\}$ for the process obtained from P by capture avoiding substitution of x for y in P. Structural congruence expresses basic identities on the structure of processes, while reduction expresses the behaviour of processes.

Definition 2.2. *Structural congruence* $(P \equiv Q)$, is the least congruence relation on processes such that

$$\begin{array}{llll}
P \mid \mathbf{0} \equiv P & (\mathrm{S}\mathbf{0}) & P \equiv_\alpha Q \Rightarrow P \equiv Q & (\mathrm{S}\alpha) \\
P \mid Q \equiv Q \mid P & (\mathrm{S|C}) & P \mid (Q \mid R) \equiv (P \mid Q) \mid R & (\mathrm{S|A}) \\
(\boldsymbol{\nu} x)\mathbf{0} \equiv \mathbf{0} & (\mathrm{S}\boldsymbol{\nu}\mathbf{0}) & x \notin fn(P) \Rightarrow P \mid (\boldsymbol{\nu} x)Q \equiv (\boldsymbol{\nu} x)(P \mid Q) & (\mathrm{S}\boldsymbol{\nu}|) \\
(\boldsymbol{\nu} x)(\boldsymbol{\nu} y)P \equiv (\boldsymbol{\nu} y)(\boldsymbol{\nu} x)P & (\mathrm{S}\boldsymbol{\nu}\boldsymbol{\nu}). & &
\end{array}$$

Definition 2.3. *Reduction* $(P \to Q)$, is the binary relation on processes defined by

$$\begin{array}{ll}
x\langle y\rangle.Q \mid x(z).P \to Q \mid P\{y/z\} & (\mathrm{RC}) \\
x\langle y\rangle.Q \mid !x(z).P \to Q \mid P\{y/z\} \mid !x(z).P & (\mathrm{R!}) \\
x.\mathtt{inl};P \mid x.\mathsf{case}(Q,R) \to P \mid Q & (\mathrm{RL}) \\
x.\mathtt{inr};P \mid x.\mathsf{case}(Q,R) \to P \mid R & (\mathrm{RR}) \\
Q \to Q' \Rightarrow P \mid Q \to P \mid Q' & (\mathrm{R|}) \\
P \to Q \Rightarrow (\boldsymbol{\nu} y)P \to (\boldsymbol{\nu} y)Q & (\mathrm{R}\boldsymbol{\nu}) \\
P \equiv P',\, P' \to Q',\, Q' \equiv Q \Rightarrow P \to Q & (\mathrm{R}\equiv)
\end{array}$$

Notice that reduction is closed (by definition) under structural congruence. Reduction specifies the computations a process performs on its own. To characterize the interactions a process may perform with its environment, we introduce a labelled transition system; the standard early transition system for the π-calculus (Sangiorgi and Walker 2001) extended with appropriate labels and transition rules for the choice constructs. A transition $P \xrightarrow{\alpha} Q$ denotes that process P may evolve to process Q by performing the action represented by the label α. Transition labels are given by

$$\alpha ::= \overline{x\langle y\rangle} \mid x(y) \mid \overline{(\nu y)x\langle y\rangle} \mid x.\mathsf{inl} \mid x.\mathsf{inr} \mid \overline{x.\mathsf{inl}} \mid \overline{x.\mathsf{inr}} \mid \boldsymbol{\tau}.$$

Actions are input $x(y)$, the left/right offers $x.\mathtt{inl}$ and $x.\mathtt{inr}$, and their matching co-actions, respectively the output $\overline{x\langle y\rangle}$ and bound output $\overline{(\boldsymbol{\nu} y)x\langle y\rangle}$ actions, and the left/right selections $\overline{x.\mathtt{inl}}$ and $\overline{x.\mathtt{inr}}$. The bound output $\overline{(\boldsymbol{\nu} y)x\langle y\rangle}$ denotes extrusion of a fresh name y along (channel) x. Internal action is denoted by $\boldsymbol{\tau}$, in general an action α $(\overline{\alpha})$ requires a matching $\overline{\alpha}$ (α) in the environment to enable progress, as specified by the transition rules. For a label α, we define the sets $fn(\alpha)$ and $bn(\alpha)$ of free and bound names, respectively, as usual. We denote by $s(\alpha)$ the subject of α (e.g. x in $x\langle y\rangle$). In our logical interpretation of sessions that is presented in the following sections we will restrict ourselves to bound output in the style of Sangiorgi (1996), we detail here a more general π-calculus for the sake of completeness.

$$\frac{P \xrightarrow{\alpha} Q}{(\nu y)P \xrightarrow{\alpha} (\nu y)Q}(\mathsf{res}) \qquad \frac{P \xrightarrow{\alpha} Q}{P \mid R \xrightarrow{\alpha} Q \mid R}(\mathsf{par}) \qquad \frac{P \xrightarrow{\overline{\alpha}} P' \quad Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}(\mathsf{com})$$

$$\frac{P \xrightarrow{\overline{(\nu y)x\langle y\rangle}} P' \quad Q \xrightarrow{x(y)} Q'}{P \mid Q \xrightarrow{\tau} (\nu y)(P' \mid Q')}\ (\mathsf{close}) \qquad \frac{P \xrightarrow{\overline{x\langle y\rangle}} Q}{(\nu y)P \xrightarrow{\overline{(\nu y)x\langle y\rangle}} Q}\ (\mathsf{open}) \qquad x\langle y\rangle.P \xrightarrow{\overline{x\langle y\rangle}} P\ (\mathsf{out})$$

$$\frac{}{x(y).P \xrightarrow{x(z)} P\{z/y\}}\ (\mathsf{in}) \qquad \frac{}{!x(y).P \xrightarrow{x(z)} P\{z/y\} \mid !x(y).P}\ (\mathsf{rep}) \qquad \frac{}{x.\mathtt{inl};P \xrightarrow{\overline{x.\mathtt{inl}}} P}\ (\mathsf{lout})$$

$$\frac{}{x.\mathtt{inr};P \xrightarrow{\overline{x.\mathtt{inr}}} P}\ (\mathsf{rout}) \qquad \frac{}{x.\mathtt{case}(P,Q) \xrightarrow{x.\mathtt{inl}} P}\ (\mathsf{lin}) \qquad \frac{}{x.\mathtt{case}(P,Q) \xrightarrow{x.\mathtt{inr}} Q}\ (\mathsf{rin})$$

Fig. 1. A π-calculus labelled transition system.

Definition 2.4 (labelled transition system). The relation *labelled transition* ($P \xrightarrow{\alpha} Q$) is defined by the rules in Figure 1, subject to the side conditions: in rule (res), we require $y \notin fn(\alpha)$; in rule (par), we require $bn(\alpha) \cap fn(R) = \emptyset$; in rule (close), we require $y \notin fn(Q)$. We omit the symmetric versions of rules (par), (com), and (close).

We recall some basic facts about reduction, structural congruence and labelled transition, namely: closure of labelled transitions under structural congruence, and coincidence of τ-labelled transition and reduction (Sangiorgi and Walker 2001). We write $\rho_1\rho_2$ for relation composition (e.g. $\xrightarrow{\tau}\equiv$).

Proposition 2.5.

1. if $P \equiv\xrightarrow{\alpha} Q$, then $P \xrightarrow{\alpha}\equiv Q$.
2. $P \to Q$ if and only if $P \xrightarrow{\tau}\equiv Q$.

3. Intuitionistic linear logic as session types

In this section, we detail our main contribution of interpreting a DILL as a session typing discipline for the process calculus of the previous section (we name this system πDILL). We assume some familiarity with linear logic and sequent calculi, but nonetheless we will introduce each connective and its process interpretation incrementally, for the sake of presentation.

We consider a sequent calculus for intuitionistic linear logic in the style of DILL (Barber 1997), augmented with a faithful proof term assignment which allows us to refer to proofs as syntactic objects. A sequent is written as $\Gamma;\Delta \vdash D : A$, denoting that D is a proof of proposition A, under the linear assumptions in Δ and the unrestricted (or exponential) assumptions in Γ. It turns out that the sequent calculus rules can be directly translated into session typing rules for processes in which the session behaviour is described by the linear proposition A. We make this correspondence explicit through the judgment $\Gamma;\Delta \vdash D \rightsquigarrow P :: z : A$, denoting that proof D maps to process P, which in turn provides a session behaviour typed by A along channel z, provided it is composed with processes implementing the session behaviours specified by Γ and Δ along the appropriate channels.

Typing is defined modulo structural congruence, as often adopted in systems for process calculi. Furthermore, we tacitly assume that all channels declared in Δ, Γ and the channel z are distinct.

As will become clear in the following sections, sequent calculus right rules correspond to rules which detail how a process can implement the session behaviour described by the considered connective. Dually, left rules explain how a process can make use of a session of a given type. Traditionally, session types are equipped with some notion of behavioural duality, in which the behaviour of the inhabitants of a type is in some sense symmetric to the behaviour of the inhabitants of its dual (e.g. the output session is dual to the input session, the choice session is dual to the branch session). In our setting a notion of behavioural duality also arises naturally from the additive and multiplicative nature of linear logic propositions.

Multiplicative conjunction $\otimes$ and implication $\multimap$ are dual in the sense that using a session of one type is equivalent to implementing a session of the other (the type $A \otimes B$ types processes that output a session of type A and proceeds as specified by B, while the type $A \multimap B$ types processes that input a session of type A and proceed as specified by B). The same applies to additive conjunction and disjunction (which correspond to branching and choice, respectively). Namely, the type $A \oplus B$ types processes that may choose either a left option of type A or a right option of type B respectively, while the type $A \multimap B$ types processes that offer a choice between both a type A behaviour and a type B behaviour. Composition of the two dual endpoints of an open session is then logically represented by the cut rule, that matches a positive occurrence of a session type with a negative occurrence of the same type, hiding the communication channel.

Throughout the following section we illustrate our type system with a simple example, typically used to motivate session based process interactions (see, e.g. Gay and Hole (2005)), involving a server that offers a buy and a quote operation and the respective client.

3.1. *Linear cut*

In logic, the cut rule allows us to reason using lemmas. A proof of C (the theorem) is well formed if it is obtained by the composition of a proof of C under the assumption of A (the lemma) and a proof of A. In linear logic, the cut rule is written as

$$\frac{\Gamma ; \Delta \vdash D : A \quad \Gamma ; \Delta', x{:}A \vdash E : C}{\Gamma ; \Delta, \Delta' \vdash \mathsf{cut}\ D\ (x.\,E) : C}\ \mathsf{cut}.$$

In essence, cut allows us to compose two proofs – one providing A and the other one using A to provide C. This principle of composition is captured in the process interpretation as follows:

$$\frac{\Gamma ; \Delta \vdash D \rightsquigarrow P :: x : A \quad \Gamma ; \Delta', x{:}A \vdash E \rightsquigarrow Q :: z : C}{\Gamma ; \Delta, \Delta' \vdash \mathsf{cut}\ D\ (x.\,E) \rightsquigarrow (\nu x)(P \mid Q) :: z : C}\ T\mathsf{cut}.$$

The process P implements session A along channel x, while process Q implements session C along channel z, under the assumption that a session of type A is available on x. Furthermore, since we follow a linear typing discipline, Q requires *all* the behaviour supplied by P along x and therefore composing the two processes must necessarily restrict the scope of x to the two processes.

This identification of cut with typed composition is not arbitrary, and turns out to be much deeper than it might first seem. The point of composing two processes is for them to interact with each other. Generally, both P and Q may interact with the 'external' process environment (captured by Δ and Δ', respectively), but the interesting interactions are those in which both P and Q communicate with each other and evolve together to some residual processes P' and Q'. All these interactions (both with the environment and between the two processes) can be understood proof theoretically through the process of cut elimination in a proof (called interchangeably cut reduction). Throughout this development, we take the correspondence of principle cut reductions (when a right rule is cut with the corresponding left rule) and process reductions as the guiding principle in our design, in the same manner as the correspondence between proof reductions and λ-term reductions guide the Curry–Howard isomorphism. The interpretation of the logical cut as composition-plus-hiding over process behaviours was introduced by Abramsky (1993), even if in a simpler setting of CSP style trace models.

3.2. *Linear implication*

Implication in linear logic, written $A \multimap B$, is commonly understood as a proof transformation process: provide me with exactly one proof of A and I shall make full use of it to produce exactly one proof of B. Dually, using $A \multimap B$ requires us to exhibit a proof of A, which then warrants the use of a proof of B. The rules for implication are:

$$\frac{\Gamma; \Delta, x : A \vdash D : B}{\Gamma; \Delta \vdash \multimap\mathsf{R}\ (x.D) : A \multimap B}\ {\multimap}\mathsf{R} \qquad \frac{\Gamma; \Delta_1 \vdash E_1 : A \quad \Gamma; \Delta_2, x : B \vdash E_2 : C}{\Gamma; \Delta_1, \Delta_2, x : A \multimap B \vdash \multimap\mathsf{L}\ x\ E_1\ (x.E_2) : C}\ {\multimap}\mathsf{L}.$$

We can also interpret the implication $A \multimap B$ as an object that *inputs* A and then produces a B. Using such an object therefore requires an output of A which then allows for the use of B. Thus, the process interpretation for implication is as input on the right and output on the left, as follows:

$$\frac{\Gamma; \Delta, x : A \vdash D \rightsquigarrow P :: z : B}{\Gamma; \Delta \vdash \multimap\mathsf{R}\ (x.D) \rightsquigarrow z(x).P :: z : A \multimap B}\ T{\multimap}\mathsf{R}$$

$$\frac{\Gamma; \Delta_1 \vdash E_1 \rightsquigarrow P :: y : A \quad \Gamma; \Delta_2, x : B \vdash E_2 \rightsquigarrow Q :: z : C}{\Gamma; \Delta_1, \Delta_2, x : A \multimap B \vdash \multimap\mathsf{L}\ x\ E_1\ (x.E_2) \rightsquigarrow (\nu y)x\langle y \rangle.(P \mid Q) :: z : C}\ T{\multimap}\mathsf{L}.$$

Note how in the left rule, we output a fresh name y, on which the process P implements the session behaviour A. The fresh output, combined with the context splitting ensures that Q does not interfere with P in any way. Throughout our development we will restrict ourselves to outputs of fresh names, in the style of Sangiorgi (1996).

As mentioned in the previous section, we can validate our correspondence by considering the principle cut-elimination steps for linear implication, which are given below in proof term form ($\Rightarrow$ stands for cut reduction and $\hat{D}^x$ stands for the process that corresponds to proof term D, which depends on name x):

$$\begin{array}{ll}
\mathsf{cut}\ (\multimap\mathsf{R}\ (y.\,D_y))\ (x.\,\multimap\mathsf{L}\ x\ E_1\ (x.\,E_{2x})) & \rightsquigarrow\ (\nu x)((x(y).\,\hat{D}^x) \mid (\nu y)\,x\langle y\rangle.\,(\hat{E}_1^y \mid \hat{E}_2^z)) \\
\Rightarrow & \qquad \rightarrow \\
\mathsf{cut}\ (\mathsf{cut}\ E_1\ (y.\,D_y))\ (x.\,E_{2x}) & \rightsquigarrow\ (\nu x)(\nu y)(\hat{D}^x \mid \hat{E}_1^y \mid \hat{E}_2^z).
\end{array}$$

3.3. *Multiplicative unit and conjunction*

The multiplicative unit of intuitionistic linear logic, written $\mathbf{1}$, is a proposition that is shown using no resources. Dually, we use $\mathbf{1}$ by just silently consuming it. The sequent calculus rules for this connective are

$$\frac{}{\Gamma;\cdot\vdash \mathbf{1}\mathsf{R} : \mathbf{1}}\ \mathbf{1}\mathsf{R} \qquad \frac{\Gamma;\Delta\vdash D : C}{\Gamma;\Delta, x:\mathbf{1}\vdash \mathbf{1}\mathsf{L}\ x\ D : C}\ \mathbf{1}\mathsf{L}.$$

The process interpretation for the multiplicative unit is the terminated session, or the inactive process:

$$\frac{}{\Gamma;\cdot\vdash \mathbf{1}\mathsf{R} \rightsquigarrow \mathbf{0} :: z : \mathbf{1}}\ T\mathbf{1}\mathsf{R} \qquad \frac{\Gamma;\Delta\vdash D \rightsquigarrow P\ : C}{\Gamma;\Delta, x:\mathbf{1}\vdash \mathbf{1}\mathsf{L}\ x\ D \rightsquigarrow P\ : C}\ T\mathbf{1}\mathsf{L}.$$

The intuition is that we provide a session of type $\mathbf{1}$ with the terminated process (no further ambient resources can be used), and 'use it' by simply erasing. This is one of the two cases where no process reduction actually takes place in composition, since the inactive process and the scope restriction are erased through *structural congruence*, not through reduction.

Multiplicative conjunction, written $A \otimes B$, requires us to split our resources in order to provide both an A and a B. Using such a proposition simply adds A and B to the context

$$\frac{\Gamma;\Delta\vdash D_1 : A \quad \Gamma;\Delta'\vdash D_2 : B}{\Gamma;\Delta,\Delta'\vdash \otimes\mathsf{R}\ D_1\ D_2 : A\otimes B}\ \otimes\mathsf{R} \qquad \frac{\Gamma;\Delta, y:A, x:B\vdash E : C}{\Gamma;\Delta, x:A\otimes B\vdash \otimes\mathsf{L}\ x\ (y.x.\,E) : C}\ \otimes\mathsf{L}.$$

The process interpretation for $\otimes$ is the exact behaviour dual of $\multimap$. While the right rule for $\multimap$ corresponds to input and the left rule corresponds to output, the right and left rules for $\otimes$ correspond to output and input, respectively:

$$\frac{\Gamma;\Delta\vdash D_1 \rightsquigarrow P :: y : A \quad \Gamma;\Delta'\vdash D_2 \rightsquigarrow Q :: z : B}{\Gamma;\Delta,\Delta'\vdash \otimes\mathsf{R}\ D_1\ D_2 \rightsquigarrow (\nu y)z\langle y\rangle.(P \mid Q) :: z : A\otimes B}\ T\otimes\mathsf{R}$$

$$\frac{\Gamma;\Delta, y:A, x:B\vdash E \rightsquigarrow P :: z : C}{\Gamma;\Delta, x:A\otimes B\vdash \otimes\mathsf{L}\ x\ (y.x.\,E) \rightsquigarrow x(y).P :: z : C}\ T\otimes\mathsf{L}.$$

Notice how in the right rule for $\otimes$ we create a fresh name y, along which the session of type A is offered by process P, while B is offered along the residual channel z, by process Q. The left rule simply inputs along x, binding the input name to y (which offers session A) in the continuation P, where x now offers B. The proof reductions that validate this interpretation (as well as the corresponding process reductions) are given below:

$$\begin{array}{ll}
\mathsf{cut}\ (\otimes\mathsf{R}\ D_1\ D_2)\ (x.\,\otimes\mathsf{L}\ x\ (y.x.\,E_{xy})) & \rightsquigarrow\ (\nu x)(((\nu y)\,x\langle y\rangle.\,(\hat{D}_1^y \mid \hat{D}_2^x)) \mid x(y).\,\hat{E}^z) \\
\Rightarrow & \qquad \rightarrow \\
\mathsf{cut}\ D_1\ (y.\,\mathsf{cut}\ D_2\ (x.\,E_{xy})) & \rightsquigarrow\ (\nu x)(\nu y)(\hat{D}_1^y \mid \hat{D}_2^x \mid \hat{E}^z).
\end{array}$$

3.3.1. *Example.* We now consider a simple example that illustrates the connectives introduced this far. We model a client process that wishes to perform a 'buy' operation on a remote server. The client does so by sending to the server a product name and a credit card number, after which it receives back a receipt. From the client perspective, the session protocol exposed by the server can be specified by the following type:

$$\mathsf{ServerProto} \triangleq N \multimap I \multimap (N \otimes \mathbf{1}).$$

We assume that, N and I are types representing shareable values such as strings and integers. To simplify, we set $N = I = \mathbf{1}$ (an extension of this system with basic and dependent data types is given in Toninho *et al.* (2011)). Assuming s to be the name of the session channel along which the client and the server interact, the following process implements the client:

$$\mathsf{BClntBody}_s \triangleq (\boldsymbol{\nu} tea)s\langle tea\rangle.(\boldsymbol{\nu} cc)s\langle cc\rangle.s(r).\mathbf{0}$$

The process above specifies a client that buys tea from the server (we abstract away what the client does with the receipt from the server). First it sends the identification of the product to the server, then its credit card information and finally receives the appropriate receipt. We then have that the following is derivable:

$$\cdot; s : \mathsf{ServerProto} \vdash \mathsf{BClntBody}_s :: - : \mathbf{1}$$

We write $-$ for an anonymous variable that does not appear in the typed process. This is possible since the inactive process $\mathbf{0}$ is typed by $x : \mathbf{1}$ and does not make use of x. The server code is as follows:

$$\mathsf{SrvBody}_s \triangleq s(pn).s(cn)(\boldsymbol{\nu} rc)s\langle rc\rangle.\mathbf{0}$$

It is straightforward to see that $\cdot;\cdot \vdash \mathsf{SrvBody}_s :: s : \mathsf{ServerProto}$ is derivable. By composing the two processes with a cut, we obtain the following:

$$\cdot;\cdot \vdash (\boldsymbol{\nu} s)(\mathsf{SrvBody}_s \mid \mathsf{BClntBody}_s) :: - : \mathbf{1}$$

In this simple example, we have only introduced processes that interact along a single session. However, our system accommodates the full generality of binary session types (e.g. a process interacting along multiple sessions is $x : A \multimap \mathbf{1}, y : A \otimes \mathbf{1} \vdash y(w).(\boldsymbol{\nu} k)x\langle k\rangle.\mathbf{0} :: - : \mathbf{1}$).

3.4. *Additive conjunction and disjunction*

We now consider additive conjunction $A \& B$ and disjunction $A \oplus B$. Additive conjunction represents alternative availability of resources (i.e. we can provide A and B, but only one of them).

$$\frac{\Gamma;\Delta \vdash D_1 : A \quad \Gamma;\Delta \vdash D_2 : B}{\Gamma;\Delta \vdash \&\mathsf{R}\; D_1\; D_2 : A \,\&\, B}\;\&\mathsf{R} \qquad \frac{\Gamma;\Delta, x : A \vdash E : C}{\Gamma;\Delta, x : A \,\&\, B \vdash \&\mathsf{L}_1\; x\; (x.E) : C}\;\&\mathsf{L}_1$$

$$\frac{\Gamma;\Delta, x : B \vdash E : C}{\Gamma;\Delta, x : A \,\&\, B \vdash \&\mathsf{L}_2\; x\; (x.E) : C}\;\&\mathsf{L}_2.$$

The process interpretation of $\&$ is a form of (binary) branching. A process implementing a session of type $A \& B$ offers the alternative between a session of type A and one of type B. Using such a session entails selecting the appropriate alternative (either inl or inr):

$$\frac{\Gamma; D \vdash D_1 \rightsquigarrow P :: z : A \quad \Gamma; \Delta \vdash D_2 \rightsquigarrow Q :: z : B}{\Gamma; \Delta \vdash \&\mathsf{R}\ D_1\ D_2 \rightsquigarrow z.\mathsf{case}(P_1, P_2) :: z : A \& B}\ T\&\mathsf{R}$$

$$\frac{\Gamma; \Delta, x : A \vdash E \rightsquigarrow Q :: z : C}{\Gamma; \Delta, x : A \& B \vdash \&\mathsf{L}_1\ x\ (x. E) \rightsquigarrow x.\mathsf{inl}; Q :: z : C}\ T\&\mathsf{L}_1$$

$$\frac{\Gamma; \Delta, x : B \vdash E \rightsquigarrow Q :: z : C}{\Gamma; \Delta, x : A \& B \vdash \&\mathsf{L}_2\ x\ (x. E) \rightsquigarrow x.\mathsf{inr}; Q :: z : C}\ T\&\mathsf{L}_2.$$

The proof and process reductions that validate this interpretation are as follows (for the sake of presentation, we omit the proof reduction for the second left rules since they are identical):

$$\begin{array}{lcl} \mathsf{cut}\ (\&\mathsf{R}\ D_1\ D_2)\ (x.\&\mathsf{L}_1\ x\ (x. E_x)) & \rightsquigarrow & (\nu x)(x.\mathsf{case}(\hat{D}_1^x, \hat{D}_2^x) \mid x.\mathsf{inl}; \hat{E}^z) \\ \Rightarrow & & \rightarrow \\ \mathsf{cut}\ D_1\ (x. E_x) & \rightsquigarrow & (\nu x)(\hat{D}_1^x \mid \hat{E}^z). \end{array}$$

Additive disjunction is the dual of additive conjunction. While additive conjunction represents alternative availability of resources (i.e. both resources are available to the client, and he chooses which to use), the additive disjunction $A \oplus B$ represents alternative availability of resources in which the choice is made by the one supplying the resources, that is, the client does not know *a priori* if it is A or B that is actually available (and hence needs to branch on the two possibilities):

$$\frac{\Gamma; \Delta \vdash D : A}{\Gamma; \Delta \vdash \oplus\mathsf{R}_1\ D : A \oplus B}\ \oplus\mathsf{R}_1 \qquad \frac{\Gamma; \Delta \vdash D : B}{\Gamma; \Delta \vdash \oplus\mathsf{R}_2\ D : A \oplus B}\ \oplus\mathsf{R}_2$$

$$\frac{\Gamma; \Delta, x : A \vdash E_1 : C \quad \Gamma; \Delta \vdash x : B \vdash E_2 : C}{\Gamma; \Delta, x : A \oplus B \vdash \oplus\mathsf{L}\ x\ (x. E_1)\ (x. E_2) : C}\ \oplus\mathsf{L}.$$

The process interpretation captures the duality mentioned above in a precise sense. The right rules for $\oplus$ correspond to the left rules for additive conjunction (either a choice of inl or inr, respectively), while the left rule for $\oplus$ corresponds to the right rule (a case analysis):

$$\frac{\Gamma; \Delta \vdash D \rightsquigarrow P :: z : A}{\Gamma; \Delta \vdash \oplus\mathsf{R}_1\ D \rightsquigarrow z.\mathsf{inr}; P :: z : A \oplus B}\ T\oplus\mathsf{R}_1$$

$$\frac{\Gamma; \Delta \vdash D \rightsquigarrow P :: z : B}{\Gamma; \Delta \vdash \oplus\mathsf{R}_2\ D \rightsquigarrow z.\mathsf{inr}; P :: z : A \oplus B}\ T\oplus\mathsf{R}_2$$

$$\frac{\Gamma; \Delta, x : A \vdash E_1 \rightsquigarrow Q_1 :: z : C \quad \Gamma; \Delta, x : B \vdash E_2 \rightsquigarrow Q_2 :: z : C}{\Gamma; \Delta, x : A \oplus B \vdash \oplus\mathsf{L}\ x\ (x. E_1)\ (x. E_2) \rightsquigarrow x.\mathsf{case}(Q_1, Q_2) :: z : C}\ T\oplus\mathsf{L}.$$

Similarly, we obtain the following proof and process reductions (we show only the reductions for the first right rule, the reductions for the remaining one are identical):

$$\begin{array}{lcl} \mathsf{cut}\ (\oplus\mathsf{R}_1\ D)\ (x.\oplus\mathsf{L}\ x\ (x. E_{1x})\ (x. E_{2x})) & \rightsquigarrow & (\nu x)(x.\mathsf{inl}; \hat{D}^x \mid x.\mathsf{case}(\hat{E}_1^z, \hat{E}_2^z)) \\ \Rightarrow & & \rightarrow \\ \mathsf{cut}\ D\ (x. E_{1x}) & \rightsquigarrow & (\nu x)(\hat{D}^x \mid \hat{E}_1^z). \end{array}$$

3.4.1. *Extending the example.* We can now easily extend our earlier example of the client and server to include branching. Consider the following type for the server interface:

$$\mathsf{ServerProto2} \triangleq (N \multimap I \multimap (N \otimes \mathbf{1})) \mathbin{\&} (N \multimap (I \otimes \mathbf{1})).$$

The type above models a server that offers the 'buy' operation from before, but also a 'quote' operation in which the client sends a product name and the server replies with the price for the respective product. The ability to offer multiple services is modelled with the additive conjunction. The client code from before can be easily extended by first choosing the appropriate operation:

$$\mathsf{BClntBody2}_s \triangleq s.\mathsf{inl}; (\boldsymbol{\nu} tea)s\langle tea\rangle.(\boldsymbol{\nu} cc)s\langle cc\rangle.s(r).\mathbf{0}$$

The server code is now extended with the appropriate communication steps:

$$\mathsf{SrvBody2}_s \triangleq s.\mathsf{case}(s(pn).s(cn)(\boldsymbol{\nu} rc)s\langle rc\rangle.\mathbf{0}, s(pn).(\boldsymbol{\nu} pr)s\langle pr\rangle.\mathbf{0})$$

It is straightforward to see that both the server and client processes have the appropriate types, and we thus obtain the following composed system through an instance of cut:

$$\frac{\cdot;\cdot \vdash \mathsf{SrvBody2}_s :: s : \mathsf{ServerProto2} \qquad \cdot; s : \mathsf{ServerProto2} \vdash \mathsf{BClntBody2}_s :: - : \mathbf{1}}{\cdot;\cdot \vdash (\boldsymbol{\nu} s)(\mathsf{SrvBody2}_s \mid \mathsf{BClntBody2}_s) :: - : \mathbf{1}}$$

3.5. *Exponential*

The linear logic exponential $!A$ enables a form of controlled weakening and contraction. A proposition $!A$ provides an arbitrary number of copies of the resource A, including 0. To prove $!A$, the usage of linear resources is therefore disallowed (otherwise this would limit the number of times A can be used):

$$\frac{\Gamma; \cdot \vdash D : A}{\Gamma; \cdot \vdash \mathsf{!R}\ D : !A}\ \mathsf{!R}$$

Using a proof of $!A$ simply moves the proposition A to the unrestricted context, accurately capturing the fact that it can be used arbitrarily often. However, to actually be able to make use of A, one must explicitly *copy* it to the linear context (this rule is hence called copy):

$$\frac{\Gamma, u : A; \Delta \vdash D : C}{\Gamma; \Delta, x : !A \vdash \mathsf{!L}\ x\ (u.D) : C}\ \mathsf{!L} \qquad \frac{\Gamma, u : A; \Delta, y : A \vdash D : C}{\Gamma, u : A; \Delta \vdash \mathsf{copy}\ u\ (y.D) : C}\ \mathsf{copy}$$

The process interpretation of the exponential is a slightly more subtle, albeit not surprising. Given that $!A$ denotes the ability to offer A arbitrarily often, a process that implements this behaviour consists of a replicated process implementing a session of type A

$$\frac{\Gamma; \cdot \vdash D \leadsto P :: y : A}{\Gamma; \cdot \vdash \mathsf{!R}\ D \leadsto !z(y).P :: z : !A}\ T\mathsf{!R}$$

We consider only a guarded form of replicated inputs. The process in the above rule waits for an input along z, after which it will spawn a copy of P that will offer the appropriate session along the channel that was initially input. The left process rule for ! is silent, since it does not actually make use of the replicated process. The actual action takes place in the copy rule, where the replicated input is matched by a corresponding (fresh) output:

$$\frac{\Gamma, u : A; \Delta \vdash E \rightsquigarrow Q :: z : C}{\Gamma; \Delta, x : !A \vdash \mathsf{!L}\ x\ E \rightsquigarrow Q\{x/u\} :: z : C}\ T\mathsf{!L}$$

$$\frac{\Gamma, u : A; \Delta, y : A \vdash D \rightsquigarrow P :: z : C}{\Gamma, u : A; \Delta \vdash \mathsf{copy}\ u\ (y.D) \rightsquigarrow (\nu y)u\langle y\rangle.P :: z : C}\ T\mathsf{copy}$$

The use of unrestricted hypotheses requires an additional cut principle, in which the cut formula is added to the unrestricted context (and the linear context is empty in the first premise)

$$\frac{\Gamma; \cdot \vdash D : A \quad \Gamma, u : A; \Delta \vdash E : C}{\Gamma; \Delta \vdash \mathsf{cut}^!\ D\ (u.E) : C}\ \mathsf{cut}^!$$

Just as linear cut corresponds to composition of linear resources, the exponential cut allows for composition of unrestricted resources

$$\frac{\Gamma; \cdot \vdash D \rightsquigarrow P :: y : A \quad \Gamma, u : A; \Delta \vdash E \rightsquigarrow Q :: z : C}{\Gamma; \Delta \vdash \mathsf{cut}^!\ D\ (u.E) \rightsquigarrow (\nu u)(!u(y).P \mid Q) :: z : C}\ \mathsf{cut}^!.$$

Similar to the case for **1**, the principle (linear) cut for ! does not map to a process reduction. Proof theoretically, such a cut reduces to an instance of $\mathsf{cut}^!$, for which the following reduction is obtained, whenever an instance of copy is reached:

$$\begin{array}{lcl}\mathsf{cut}^!\ D\ (u.\mathsf{copy}\ u\ (y.\ E_{uy})) & \rightsquigarrow & (\nu u)((!u(y).\hat{D}^y) \mid (\nu y)\,u\langle y\rangle.\,\hat{E}^z) \\ \Rightarrow & & \rightarrow \\ \mathsf{cut}\ D\ (y.\mathsf{cut}^!\ D\ (u.\ E_{uy})) & \rightsquigarrow & (\nu y)(\hat{D}^y \mid (\nu u)((!u(y).\,\hat{D}^y) \mid \hat{E}^z)).\end{array}$$

3.5.1. *Replicating the example.* We now elaborate on our running example, in order to illustrate sharing and session initiation. Consider now a different client, that picks the 'quote' rather than the 'buy' operation, and the corresponding composed system.

$$\begin{array}{ll}\mathsf{QCIntBody}_s & \triangleq s.\mathtt{inr}; (\boldsymbol{\nu} cof)s\langle cof\rangle.s(pr).\mathbf{0} \\ \mathsf{QSimple} & \triangleq (\boldsymbol{\nu} s)(\mathsf{SrvBody2}_s \mid \mathsf{QCIntBody}_s).\end{array}$$

We have the typings $\cdot; s{:}\mathsf{ServerProto2} \vdash \mathsf{QCIntBody}_s :: -{:}\mathbf{1}$ and $\cdot;\cdot \vdash \mathsf{QSimple} :: -{:}\mathbf{1}$.

In these examples, there is a single installed pair client-server, where the session is already initiated, and only known to the two partners. To illustrate sharing, we now consider a replicated server. Such a replicated server is able to spawn a fresh session instance for each initial invocation, each one conforming to the general behaviour specified by ServerProto2, and can be typed by !ServerProto2. Correspondingly, clients must initially invoke the replicated server to instantiate a new session (cf. the Tcopy rule).

$$\begin{array}{ll}\mathsf{QClient} & \triangleq (\boldsymbol{\nu} s)c\langle s\rangle.\mathsf{QCIntBody}_s \\ \mathsf{BClient} & \triangleq (\boldsymbol{\nu} s)c\langle s\rangle.\mathsf{BCIntBody2}_s \\ \mathsf{Server} & \triangleq !c(s).\mathsf{SrvBody2}_s \\ \mathsf{SharSys} & \triangleq (\boldsymbol{\nu} c)(\mathsf{Server} \mid \mathsf{BClient} \mid \mathsf{QClient})\end{array}$$

For the shared server, by T!R, we type $\cdot;\cdot \vdash \mathsf{Server} :: c{:}!\mathsf{ServerProto2}$. We also have, for the clients, by Tcopy the typings $c{:}\mathsf{ServerProto2}\ ;\ \cdot \vdash \mathsf{BClient} :: -{:}\mathbf{1}$ and

$c{:}\mathsf{ServerProto2}\ ;\ \cdot \vdash \mathsf{QClient} :: -{:}\mathbf{1}$. By T!L and Tcut we obtain the intended typing for the whole system: $\cdot;\cdot \vdash \mathsf{SharSys} :: - : \mathbf{1}$. Notice how the session instantiation protocol is naturally explained by the logical interpretation of the ! operator.

3.6. *Identity*

In proof theory, the identity theorem for a sequent calculus presentation of logic entails the internal completeness of logic: one can always prove an assumed proposition. Just as the computational content of cut elimination corresponds to process reduction, the computational content of the identity theorem yields a form of expansion.

Proposition 3.1. For any type A and distinct names x, y, there is a process $id_A(x, y)$ and a cut-free derivation D such that $\cdot; x : A \vdash D \rightsquigarrow id_A(x, y) :: y : A$.

The identity process $id_A(x, y)$, containing exactly the free names x, y, implements a synchronous mediator that carries out the communication protocol specified by the type A between the two channels x and y. To clarify, we analyse the interpretation of the sequent $A \otimes B \vdash B \otimes A$ as follows:

$$x : A \otimes B \vdash F \rightsquigarrow x(z).(\nu n)y\langle n\rangle.(id_B(x, n) \mid id_A(z, y)) :: y : B \otimes A$$

where $F = \otimes\mathsf{L}\ x\ (z.x.\otimes\mathsf{R}\ D\ E)$, $D \rightsquigarrow id_B(x, n)$ and $E \rightsquigarrow id_A(z, y)$. The process given above coerces a session of type $A \otimes B$ on channel x to one of type $B \otimes A$ on y by first inputting a session of type A (bound to z) and afterwards sending on y a session of type B (carried out by coercing the continuation of x to n), after which it progresses with a session of type A along y (by coercing the continuation of z to y).

3.7. *Summary*

In this section, we summarize the contributions of this section. Specifically, we present here the complete type system πDILL that arises from our interpretation of intuitionistic linear logic, which for the sake of presentation we developed incrementally throughout this section. The rules are exactly those presented before, obtained by erasing the sequent calculus proof terms, and are given in Figure 2 (where T stands for $z : C$).

The extraction $\rightsquigarrow$ of well-typed processes from sequent calculus proof terms is summarized in Figure 3. Extraction is unique up to structural congruence, since typing is by definition closed under $\equiv$.

Finally, we summarize the several proof conversions and their correspondent process reductions or equivalences. As detailed throughout the previous sections, process reductions correspond to computational proof conversions (Figure 4). The structural conversions in Figure 5 correspond to structural equivalences in the π-calculus, since they just change the order of cuts, e.g. $(\mathsf{cut}/-/\mathsf{cut}_1)$ translates to

$$(\nu x)(\hat{D}^x \mid (\nu y)(\hat{E}^y \mid \hat{F}^z)) \equiv (\nu y)((\nu x)(\hat{D}^x \mid \hat{E}^y) \mid \hat{F}^z).$$

In addition, we have two special conversions. Among those, $(\mathsf{cut}/\mathbf{1}\mathsf{R}/\mathbf{1}\mathsf{L})$ is not needed in order to simulate the π-calculus reduction, while $(\mathsf{cut}/!\mathsf{R}/!\mathsf{L})$ is. In cut-elimination

$$\frac{\Gamma;\Delta\vdash P :: T}{\Gamma;\Delta,x{:}\mathbf{1}\vdash P :: T}\ (\mathrm{T}\mathbf{1}\mathrm{L}) \qquad \frac{}{\Gamma;\cdot\vdash \mathbf{0} :: x{:}\mathbf{1}}\ (\mathrm{T}\mathbf{1}\mathrm{R})$$

$$\frac{\Gamma;\Delta,y{:}A,x{:}B\vdash P :: T}{\Gamma;\Delta,x{:}A\otimes B\vdash x(y).P :: T}\ (\mathrm{T}{\otimes}\mathrm{L}) \qquad \frac{\Gamma;\Delta\vdash P :: y{:}A \quad \Gamma;\Delta'\vdash Q :: x{:}B}{\Gamma;\Delta,\Delta'\vdash (\nu y)x\langle y\rangle.(P\mid Q) :: x{:}A\otimes B}\ (\mathrm{T}{\otimes}\mathrm{R})$$

$$\frac{\Gamma;\Delta\vdash P :: y{:}A \quad \Gamma;\Delta',x{:}B\vdash Q :: T}{\Gamma;\Delta,\Delta',x{:}A\multimap B\vdash (\nu y)x\langle y\rangle.(P\mid Q) :: T}\ (\mathrm{T}{\multimap}\mathrm{L}) \qquad \frac{\Gamma;\Delta,y{:}A\vdash P :: x{:}B}{\Gamma;\Delta\vdash x(y).P :: x{:}A\multimap B}\ (\mathrm{T}{\multimap}\mathrm{R})$$

$$\frac{\Gamma;\Delta\vdash P :: x{:}A \quad \Gamma;\Delta',x{:}A\vdash Q :: T}{\Gamma;\Delta,\Delta'\vdash (\nu x)(P\mid Q) :: T}\ (\mathsf{Tcut}) \qquad \frac{\Gamma;\cdot\vdash P :: y{:}A \quad \Gamma,u{:}A;\Delta\vdash Q :: T}{\Gamma;\Delta\vdash (\nu u)(!u(y).P\mid Q) :: T}\ (\mathsf{Tcut}^{!})$$

$$\frac{\Gamma,u{:}A;\Delta,y{:}A\vdash P :: T}{\Gamma,u{:}A;\Delta\vdash (\nu y)u\langle y\rangle.P :: T}\ (\mathsf{Tcopy})$$

$$\frac{\Gamma,u{:}A;\Delta\vdash P :: T}{\Gamma;\Delta,x{:}!A\vdash P\{x/u\} :: T}\ (\mathrm{T!L}) \qquad \frac{\Gamma;\cdot\vdash Q :: y{:}A}{\Gamma;\cdot\vdash\, !x(y).Q :: x{:}!A}\ (\mathrm{T!R})$$

$$\frac{\Gamma;\Delta,x{:}A\vdash P :: T \quad \Gamma;\Delta,x{:}B\vdash Q :: T}{\Gamma;\Delta,x{:}A\oplus B\vdash x.\mathtt{case}(P,Q) :: T}\ (\mathrm{T}{\oplus}\mathrm{L}) \qquad \frac{\Gamma;\Delta,x{:}B\vdash P :: T}{\Gamma;\Delta,x{:}A\,\&\,B\vdash x.\mathtt{inr};P :: T}\ (\mathrm{T}\&\mathrm{L}_2)$$

$$\frac{\Gamma;\Delta\vdash P :: x{:}A \quad \Gamma;\Delta\vdash Q :: x{:}B}{\Gamma;\Delta\vdash x.\mathtt{case}(P,Q) :: x{:}A\,\&\,B}\ (\mathrm{T}\&\mathrm{R}) \qquad \frac{\Gamma;\Delta,x{:}A\vdash P :: T}{\Gamma;\Delta,x{:}A\,\&\,B\vdash x.\mathtt{inl};P :: T}\ (\mathrm{T}\&\mathrm{L}_1)$$

$$\frac{\Gamma;\Delta\vdash P :: x{:}A}{\Gamma;\Delta\vdash x.\mathtt{inl};P :: x{:}A\oplus B}\ (\mathrm{T}{\oplus}\mathrm{R}_1) \qquad \frac{\Gamma;\Delta\vdash P :: x{:}B}{\Gamma;\Delta\vdash x.\mathtt{inr};P :: x{:}A\oplus B}\ (\mathrm{T}{\oplus}\mathrm{R}_2).$$

Fig. 2. The type system πDILL.

D	$\rightsquigarrow\ \hat{D}^z$	D	$\rightsquigarrow\ \hat{D}^z$
$\mathbf{1}\mathsf{R}$	$\rightsquigarrow\ 0$	$\oplus\mathsf{R}_1\ D$	$\rightsquigarrow\ z.\mathsf{inl};\hat{D}^z$
$\mathbf{1}\mathsf{L}\ x\ D$	$\rightsquigarrow\ \hat{D}^z$	$\oplus\mathsf{R}_2\ E$	$\rightsquigarrow\ z.\mathsf{inr};\hat{E}^z$
$\otimes\mathsf{R}\ D\ E$	$\rightsquigarrow\ (\nu y)\,z\langle y\rangle.\,(\hat{D}^y\mid\hat{E}^z)$	$\oplus\mathsf{L}\ x\ (x.\,D)\ (x.\,E)$	$\rightsquigarrow\ x.\mathsf{case}(\hat{D}^z,\hat{E}^z)$
$\otimes\mathsf{L}\ x\ (y.x.\,D)$	$\rightsquigarrow\ x(y).\,\hat{D}^z$	$\mathsf{cut}\ D\ (x.\,E)$	$\rightsquigarrow\ (\nu x)(\hat{D}^x\mid\hat{E}^z)$
$\multimap\mathsf{R}\ (y.\,D)$	$\rightsquigarrow\ z(y).\,\hat{D}^z$		
$\multimap\mathsf{L}\ x\ D\ (x.\,E)$	$\rightsquigarrow\ (\nu y)\,x\langle y\rangle.\,(\hat{D}^y\mid\hat{E}^z)$	$!\mathsf{R}\ D$	$\rightsquigarrow\ !z(y).\,\hat{D}^y$
$\&\mathsf{R}\ D\ E$	$\rightsquigarrow\ z.\,\mathsf{case}(\hat{D}^z,\hat{E}^z)$	$!\mathsf{L}\ x\ (u.\,D)$	$\rightsquigarrow\ \hat{D}^z\{x/u\}$
$\&\mathsf{L}_1\ x\ (x.\,D)$	$\rightsquigarrow\ x.\,\mathsf{inl};\hat{D}^z$	$\mathsf{copy}\ u\ (y.\,D)$	$\rightsquigarrow\ (\nu y)\,u\langle y\rangle.\,\hat{D}^z$
$\&\mathsf{L}_2\ x\ (x.\,E)$	$\rightsquigarrow\ x.\,\mathsf{inr};\hat{E}^z$	$\mathsf{cut}^!\ D\ (u.\,E)$	$\rightsquigarrow\ (\nu u)((!u(y).\,\hat{D}^y)\mid\hat{E}^z)$

Fig. 3. Proof D extracts to process $\hat{D}^z$.

procedures, these are always used from left to right. Here, they are listed as equivalences because the corresponding π-calculus terms are structurally congruent. The root cause for this is that the rules $\mathbf{1}\mathsf{L}$ and $!\mathsf{L}$ are *silent*: the extracted terms in the premise and conclusion are the same, modulo renaming. The structural conversions in Figure 7 push

$$\begin{array}{lcl}
\mathsf{cut}\ (\otimes\mathsf{R}\ D_1\ D_2)\ (x.\,\otimes\mathsf{L}\ x\ (y.x.\,E_{xy})) & \rightsquigarrow & (\nu x)(((\nu y)\,x\langle y\rangle.\,(\hat{D}_1^y \mid \hat{D}_2^x)) \mid x(y).\,\hat{E}^z) \\
\Rightarrow & & \rightarrow \\
\mathsf{cut}\ D_1\ (y.\,\mathsf{cut}\ D_2\ (x.\,E_{xy})) & \rightsquigarrow & (\nu x)(\nu y)(\hat{D}_1^y \mid \hat{D}_2^x \mid \hat{E}^z) \\
\mathsf{cut}\ (\multimap\mathsf{R}\ (y.\,D_y))\ (x.\,\multimap\mathsf{L}\ x\ E_1\ (x.\,E_{2x})) & \rightsquigarrow & (\nu x)((x(y).\,\hat{D}^x) \mid (\nu y)\,x\langle y\rangle.\,(\hat{E}_1^y \mid \hat{E}_2^z)) \\
\Rightarrow & & \rightarrow \\
\mathsf{cut}\ (\mathsf{cut}\ E_1\ (y.\,D_y))\ (x.\,E_{2x}) & \rightsquigarrow & (\nu x)(\nu y)(\hat{D}^x \mid \hat{E}_1^y \mid \hat{E}_2^z) \\
\mathsf{cut}\ (\&\mathsf{R}\ D_1\ D_2)\ (x.\&\mathsf{L}_i\ x\ (x.\,E_x)) & \rightsquigarrow & (\nu x)(x.\mathsf{case}(\hat{D}_1^x, \hat{D}_2^x) \mid x.\mathsf{inl};\hat{E}^z) \\
\Rightarrow & & \rightarrow \\
\mathsf{cut}\ D_i\ (x.\,E_x) & \rightsquigarrow & (\nu x)(\hat{D}_i^x \mid \hat{E}^z) \\
\mathsf{cut}\ (\oplus\mathsf{R}_i\ D)\ (x.\oplus\mathsf{L}\ x\ (x.\,E_{1x})\ (x.\,E_{2x})) & \rightsquigarrow & (\nu x)(x.\mathsf{inl};\hat{D}^x \mid x.\mathsf{case}(\hat{E}_1^z, \hat{E}_2^z)) \\
\Rightarrow & & \rightarrow \\
\mathsf{cut}\ D\ (x.\,E_{ix}) & \rightsquigarrow & (\nu x)(\hat{D}^x \mid \hat{E}_i^z) \\
\mathsf{cut}^!\ D\ (u.\,\mathsf{copy}\ u\ (y.\ E_{uy})) & \rightsquigarrow & (\nu u)((!u(y).\,\hat{D}^y) \mid (\nu y)\,u\langle y\rangle.\,\hat{E}^z) \\
\Rightarrow & & \rightarrow \\
\mathsf{cut}\ D\ (y.\,\mathsf{cut}^!\ D\ (u.\ E_{uy})) & \rightsquigarrow & (\nu y)(\hat{D}^y \mid (\nu u)((!u(y).\,\hat{D}^y) \mid \hat{E}^z))
\end{array}$$

Fig. 4. Computational conversions.

$$\begin{array}{llcl}
(\mathsf{cut}/-/\mathsf{cut}_1) & \mathsf{cut}\ D\ (x.\,\mathsf{cut}\ E_x\ (y.\ F_y)) & \equiv & \mathsf{cut}\ (\mathsf{cut}\ D\ (x.\,E_x))\ (y.\ F_y) \\
(\mathsf{cut}/-/\mathsf{cut}_2) & \mathsf{cut}\ D\ (x.\,\mathsf{cut}\ E\ (y.\ F_{xy})) & \equiv & \mathsf{cut}\ E\ (y.\,\mathsf{cut}\ D\ (x.\ F_{xy})) \\
(\mathsf{cut}/\mathsf{cut}^!/-) & \mathsf{cut}\ (\mathsf{cut}^!\ D\ (u.\,E_u))\ (x.\,F_x) & \equiv & \mathsf{cut}^!\ D\ (u.\,\mathsf{cut}\ E_u\ (x.\,F_x)) \\
(\mathsf{cut}/-/\mathsf{cut}^!) & \mathsf{cut}\ D\ (x.\,\mathsf{cut}^!\ E\ (u.\,F_{xu})) & \equiv & \mathsf{cut}^!\ E\ (u.\,\mathsf{cut}\ D\ (x.\,F_{xu})) \\
(\mathsf{cut}/\mathbf{1}\mathsf{R}/\mathbf{1}\mathsf{L}) & \mathsf{cut}\ \mathbf{1}\mathsf{R}\ (x.\,\mathbf{1}\mathsf{L}\ x\ D) & \equiv & D \\
(\mathsf{cut}/!\mathsf{R}/!\mathsf{L}) & \mathsf{cut}\ (!\mathsf{R}\ D)\ (x.\ !\mathsf{L}\ x\ (u.\,E)) & \equiv & \mathsf{cut}^!\ D\ (u.\,E)
\end{array}$$

Fig. 5. Structural conversions (I): cut conversions.

$$\begin{array}{llcl}
(\mathsf{cut}/\mathbf{1}\mathsf{L}/-) & \mathsf{cut}\ (\mathbf{1}\mathsf{L}\ y\ D)\ (x.\,F_x) & \equiv & \mathbf{1}\mathsf{L}\ y\ (\mathsf{cut}\ D\ (x.\,F_x)) \\
(\mathsf{cut}/!\mathsf{L}/-) & \mathsf{cut}\ (!\mathsf{L}\ y\ (u.\,D_u))\ (x.\,F_x) & \equiv & !\mathsf{L}\ y\ (u.\,\mathsf{cut}\ D_u\ (x.\,F_x)) \\
(\mathsf{cut}^!/-/\mathbf{1}\mathsf{L}) & \mathsf{cut}^!\ D\ (u.\,\mathbf{1}\mathsf{L}\ y\ E_u) & \equiv & \mathbf{1}\mathsf{L}\ y\ (\mathsf{cut}^!\ D\ (u.\,E_u)) \\
(\mathsf{cut}^!/-/!\mathsf{L}) & \mathsf{cut}^!\ D\ (u.\ !\mathsf{L}\ y\ (v.\,E_{uv})) & \equiv & !\mathsf{L}\ y\ (v.\ \mathsf{cut}^!\ D\ (u.\,E_{uv}))
\end{array}$$

Fig. 6. Structural conversions (II): commuting conversions.

$\mathsf{cut}^!$ into the derivation. From a proof theoretic perspective, since $\mathsf{cut}^!$ cuts a persistent variable u, $\mathsf{cut}^!$ may be duplicated or erased. On the π-calculus side, these no longer correspond to structural congruences, but, quite remarkably, to behavioural equivalences, derivable from known properties of typed processes, the (sharpened) replication theorems (Sangiorgi and Walker 2001). These hold in our system, due to our interpretation of ! types. Our operational correspondence results also depend on six commuting conversions, four in Figure 6 plus two symmetric versions. The commuting conversions push a cut up (or inside) the $\mathbf{1}$L and !L rules. During the usual cut-elimination procedures, these are used from left to right. In the correspondence with the sequent calculus, the situation is more complex. Because the $\mathbf{1}$L and !L rules do not affect the extracted term, cuts have to be permuted with these two rules in order to simulate π-calculus reduction. From the process calculus perspective, such conversions correspond to identity. There is a second group of commuting conversions (not shown), not necessary for our current development. Those do not correspond to structural congruence nor to strong bisimilarities on π-calculus,

$$\begin{array}{lll}
\mathsf{cut}^! \; D \; (u.\, \mathsf{cut} \; E_u \; (y.\; F_{uy})) & \rightsquigarrow & (\nu u)(!u(y).\hat{D}^y \mid (\nu y)(\hat{E}^y \mid \hat{F}^z)) \\
\simeq & & \simeq \\
\mathsf{cut} \; (\mathsf{cut}^! \; D \; (u.\, E_u)) \; (y.\; \mathsf{cut}^! \; D \; (u.\, F_{uy})) & \rightsquigarrow & (\nu y)((\nu u)(!u(y).\hat{D}^y \mid \hat{E}^y) \mid \\
 & & \qquad\qquad (\nu u)(!u(y).\hat{D}^y \mid \hat{F}^z)\;) \\
\mathsf{cut}^! \; D \; (u.\, \mathsf{cut}^! \; E_u \; (v.\; F_{uv})) & \rightsquigarrow & (\nu u)(!u(y).\hat{D}^y \mid (\nu v)(!v(y).\hat{E}^y \mid \hat{F}^z)) \\
\simeq & & \simeq \\
\mathsf{cut}^! \; (\mathsf{cut}^! \; D \; (u.\, E_u)) \; (v.\; \mathsf{cut}^! \; D \; (u.\, F_{uv})) & \rightsquigarrow & (\nu v)((!v(y).(\nu u)(!u(y).\hat{D}^y \mid \hat{E}^y)) \mid \\
 & & \qquad\qquad (\nu u)(!u(y).\hat{D}^y \mid \hat{F}^z)\;) \\
\mathsf{cut}^! \; (\mathsf{cut}^! \; D \; (u.\, E_u)) \; (v.\; F_v) & \rightsquigarrow & (\nu v)(!v(y).(\nu u)(!u(y).\hat{D}^y \mid \hat{E}^y)) \mid F^z) \\
\simeq & & \simeq \\
\mathsf{cut}^! \; D \; (u.\, \mathsf{cut}^! \; E_u \; (v.\; F_v)) & \rightsquigarrow & (\nu u)(!u(y).\hat{D}^y \mid (\nu v)(!v(y).\hat{E}^y \mid \hat{F}^z)) \\
\mathsf{cut}^! \; D \; (u.\, E) & \rightsquigarrow & (\nu u)(!u(y).\hat{D}^y \mid \hat{E}^z) \\
\simeq & & \simeq \\
E & \rightsquigarrow & \hat{E}^z \quad (\text{for } u \notin FN(\hat{E}^z))
\end{array}$$

Fig. 7. Structural conversions (III): cut! conversions.

as they may not preserve process behaviour in the general untyped setting, since they promote an action prefix from a subexpression to the top level. Such equations denote behavioural identities under a natural definition of typed observational congruence for our calculus (Pérez *et al.* 2012).

Definition 3.2 (relations on derivations induced by conversions). (1) $\equiv$: the least congruence on derivations generated by the structural conversions (I) and the commuting conversions (II); (2) $\simeq_s$: the least congruence on derivations generated by all structural conversions (I-III). We extend $\simeq_s$ to processes as the congruence generated by the process equations on the right. (3) $\Mapsto$: the reduction on derivations obtained by orienting all conversions in the direction shown, from left to right or top to bottom.

4. Computational correspondence, progress and preservation

We now present the results stating the key properties of our type system and logical interpretation. Theorem 4.1, states the existence of a simulation between reductions in the typed π-calculus and proof conversions / reductions, expressing a strong form of subject reduction for our type system. The proof relies on several auxiliary lemmas that relate process reduction with proof reduction at a particular type. The lemmas themselves are all very similar, so we only present the lemmas for $\otimes$ and !. The remaining lemmas, and their proofs, are detailed in Appendix A.

Lemma 4.1. Assume

1. $\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow P_1 :: x{:}C_1 \otimes C_2$ with $P_1 \xrightarrow{\overline{(\nu y)x\langle y\rangle}} P_1'$;
2. $\Gamma; \Delta_2, x{:}C_1 \otimes C_2 \vdash D_2 \rightsquigarrow Q_1 :: z{:}C$ with $Q_1 \xrightarrow{x(y)} Q_1'$.

Then,

1. $\mathsf{cut}\; D_1(x.\, D_2) \equiv\Rightarrow\equiv D$ for some D;
2. $\Gamma; \Delta_1, \Delta_2 \vdash D \leadsto Q_2 :: z : C$ for some $Q_2 \equiv (\nu x)(P_1' \mid Q_1')$.

Proof. See Appendix A.6. □

Lemma 4.2. Assume

1. $\Gamma; \Delta_1 \vdash D_1 \leadsto P_1 :: x{:}!A$ with $P_1 \overset{x(y)}{\to} P_1'$;
2. $\Gamma; \Delta_2, x{:}!A \vdash D_2 \leadsto Q_1 :: z{:}C$ with $Q_1 \overset{\overline{(\nu y)x\langle y\rangle}}{\to} Q_1'$.

Then,

1. $\mathsf{cut}\; D_1(x.\, D_2) \equiv\Rightarrow\equiv D$ for some D;
2. $\Gamma; \Delta_1, \Delta_2 \vdash D \leadsto Q_2 :: z : C$ for some $Q_2 \equiv (\nu x)(\nu y)(P_1' \mid Q_1')$.

Proof. See Appendix A.8. □

The key idea of the lemmas above is that by relating process reduction with proof reduction at a given type we can conclude a strong form of type preservation, as follows.

Theorem 4.1. Let $\Gamma; \Delta \vdash D \leadsto P :: z{:}A$ and $P \to Q$. Then there is an E such that $D \equiv\Rightarrow\equiv E$ and $\Gamma; \Delta \vdash E \leadsto Q :: z{:}A$.

Proof. See Appendix A.1. □

Theorems 4.2 and 4.3 state that any proof reduction or conversion also corresponds to either a process equivalence or to a reduction step on the π-calculus.

Theorem 4.2. Let $\Gamma; \Delta \vdash D \leadsto P :: z{:}A$ and $D \simeq_s E$. Then there is a Q where $P \simeq_s Q$ and $\Gamma; \Delta \vdash E \leadsto Q :: z{:}A$.

Proof. Following the commuting squares relating $\equiv$, $\leadsto$ and $\simeq$ in Figures 5–7. □

Theorem 4.3. Let $\Gamma; \Delta \vdash D \leadsto P :: z{:}A$ and $D \Rightarrow E$. Then there is a Q such that $P \to Q$ and $\Gamma; \Delta \vdash E \leadsto Q :: z{:}A$.

Proof. Following the commuting squares relating $\Rightarrow$, $\leadsto$ and $\to$ in Figure 4. □

Notice that the simulation of π-calculus reductions by proof term conversions provided by Theorem 4.1, and from which subject reduction follows, is very tight indeed, as reduction is simulated up to structural congruence, which is a very fine equivalence on processes. To that end, structural conversions need to be applied symmetrically (as equations), unlike in a standard proof of cut elimination, where they are usually considered as directed computational steps. Under the assumptions of Theorem 4.1, we can also prove that there is an E such that $D \;\;\Rightarrow E$ and $\Gamma; \Delta \vdash E \leadsto R :: z{:}A$, for $Q \simeq_s R$. Thus, even if one considers the proof conversions as directed reduction rules ($\Mapsto$), we still obtain a sound simulation up to typed strong behavioural congruence.

We now state type preservation and progress results for our type system. The subject reduction property (Theorem 4.4) directly follows from Theorem 4.1.

Theorem 4.4 (subject reduction). If $\Gamma;\Delta \vdash P :: z{:}A$ and $P \rightarrow Q$ then $\Gamma;\Delta \vdash Q :: z{:}A$.

Together with direct consequences of linear typing, Theorem 4.4 ensures session fidelity. Our type discipline also enforces a global progress property. For any P, define

$$\mathit{live}(P) \;\; \mathit{iff} \;\; P \equiv (\nu \overline{n})(\pi.Q \mid R) \quad \mathit{for\ some}\ \pi.Q, R, \overline{n}$$

where $\pi.Q$ is a *non-replicated* guarded process. We first establish the following contextual progress property, from which Theorem 4.5 follows as a corollary. Lemma 4.3 relies on an inversion lemma that relates types with action labels (Lemma A.1) and on a lemma that characterizes the typing of non-live processes (Lemma A.11). Note that the restriction on the contexts and on the type for the distinguished channel x in Theorem 4.5 is without loss of generality since using cut and $\mathsf{cut}^!$ we can compose arbitrary well-typed processes together and x need not occur in P due to rule $\mathbf{1}\mathsf{R}$.

Lemma 4.3. Let $\Gamma;\Delta \vdash D \rightsquigarrow P :: z{:}C$. If $\mathit{live}(P)$ then there is a Q such that either

1. $P \rightarrow Q$, or
2. $P \xrightarrow{\alpha} Q$ for α where $s(\alpha) \in (z, \Gamma, \Delta)$. More: if $C = {!A}$ for some A, then $s(\alpha) \neq z$.

Proof. See Appendix A.12 $\square$

Theorem 4.5 (progress). If $\cdot;\cdot \vdash D \rightsquigarrow P :: x{:}\mathbf{1}$ then either P is terminated, P is a composition of replicated servers or there exists Q s.t $P \rightarrow Q$.

Proof. See Appendix A.2 $\square$

5. On duality, and a classical linear logic interpretation

Although we have based our development on intuitionistic linear logic, the linear logic interpretation of session types also naturally extends to the classical setting. In this section, we briefly discuss such an alternative system, leaving a more comprehensive analysis for future investigation. The main characteristic of the classical interpretation is that it supports a full duality on the type structure. In traditional session type systems, e.g. Gay and Hole (2005), a duality relation is defined on session types, in such a way that every session type S has a unique dual $\overline{S}$, given by

$$\begin{array}{lclclcl}
\overline{\mathtt{end}} & \triangleq & \mathtt{end} & & & & \\
\overline{T?.S} & \triangleq & T!.\overline{S} & & \overline{T!.S} & \triangleq & T?.\overline{S} \\
\overline{S \oplus T} & \triangleq & \overline{S} \,\&\, \overline{T} & & \overline{S \,\&\, T} & \triangleq & \overline{S} \oplus \overline{T}.
\end{array}$$

In our case, this would correspond to

$$\begin{array}{lclclcl}
\overline{\mathbf{1}} & \triangleq & \mathbf{1} & & & & \\
\overline{T \multimap S} & \triangleq & T \otimes \overline{S} & & \overline{T \otimes S} & \triangleq & T \multimap \overline{S} \\
\overline{S \oplus T} & \triangleq & \overline{S} \,\&\, \overline{T} & & \overline{S \,\&\, T} & \triangleq & \overline{S} \oplus \overline{T}.
\end{array}$$

This duality relation does not talk about the type of shared channels. In traditional session types, the type structure is stratified, so that one distinguishes between 'session types' and 'standard types', the latter intended to type (shared) session initiation channels. In our case, shared channels are naturally typed by the exponential $!A$, without any stratification on the type structure whatsoever. Nevertheless, in our system a related notion of polarity on types is already implicitly reflected by the left/right duality of intuitionistic sequents, in the sense that we can move all channels to the left-hand side of the sequent.

Proposition 5.1. Let A be an exponential-free type.
Then $\Gamma; \Delta \vdash P :: x{:}A$ implies $\Gamma; \Delta, x{:}\overline{A} \vdash P :: -{:}\mathbf{1}$.

Proof. By induction on the structure of the given proof (see Appendix A.4). □

So, if we have $P :: x{:}A$ and $Q :: x{:}\overline{A}$ (where A has no occurrence of '!') we can compose P and Q as $(\nu x)(P \mid Q) :: -{:}\mathbf{1}$ using the above proposition and the cut rule.

A key distinctive aspects of our intuitionistic interpretation is the natural reading it offers of session typed processes as systems that implement a provided session (or service) based on a set of required sessions (or services). Another essential aspect is the asymmetry it introduces in the treatment of the exponential $!A$, whose 'dual' behaviour, in the sense discussed above, is not available in the type structure (interestingly, like in traditional session types). Intuitively, while the type of a shared server located at name s and providing protocols of type A is $!A$, and expresses the capability to **receive** (at s) an unbounded number of concurrent incoming service request messages, the dual behaviour should express the capability to **send** (to s) an unbounded number of concurrent incoming service request messages for a local endpoint satisfying the protocol $\overline{A}$.

A full duality in the context of the interpretation just sketched may be recovered by resorting to an interpretation in CLL, which assigns a dual type to every session type, in particular to $!A$, by introducing the 'why-not' exponential connective $?A$. We may then consider the following type structure

Definition 5.2 (C-types). C-types (A, B, C) are given by

$$A, B \ ::= \ \bot \mid \mathbf{1} \mid !A \mid ?A \mid A \otimes B \mid A \mathbin{⅋} B \mid A \oplus B \mid A \mathbin{\&} B.$$

The input session type $A \multimap B$ is here modelled by $\overline{A} \mathbin{⅋} B$, following the interpretation of the linear implication in CLL. The interpretation of $A \mathbin{⅋} B$ poses no problems: it types a session that inputs a channel of type A, and continues as B. It should be clear that the issues of the apparent asymmetry of $A \mathbin{⅋} B$ can be explained as we did for $A \otimes B$ in Section 3.6. Moreover, we may define a full duality on C-types, which exactly corresponds to the negation operator of CLL $(\cdot)^{\perp}$.

$$\begin{array}{rclrcl}
\overline{\mathbf{1}} & = & \bot & \overline{\bot} & = & \mathbf{1} \\
\overline{!A} & = & ?\overline{A} & \overline{?A} & = & !\overline{A} \\
\overline{A \otimes B} & = & \overline{A} \mathbin{⅋} \overline{B} & \overline{A \mathbin{⅋} B} & = & \overline{A} \otimes \overline{B} \\
\overline{A \oplus B} & = & \overline{A} \mathbin{\&} \overline{B} & \overline{A \mathbin{\&} B} & = & \overline{A} \oplus \overline{B}.
\end{array}$$

To be absolutely faithful to the classical interpretation, and without loss of expressiveness with respect to traditional session types, we split the 'session termination' type `end` into two different types: $\mathbf{1}$ and $\bot$, the units for $A \otimes B$ and $A \mathbin{⅋} B$. These may be understood as representing the session closure behaviours implemented by matching

$$\frac{}{\mathbf{0} \vdash x{:}\mathbf{1}; \Theta}\ (\mathrm{T}\mathbf{1}) \qquad \frac{P \vdash \Delta; \Theta}{P \vdash x{:}\bot, \Delta; \Theta}\ (\mathrm{T}\bot)$$

$$\frac{P \vdash \Delta, y{:}A, x{:}B; \Theta}{x(y).P \vdash \Delta, x{:}A \parr B; \Theta}\ (\mathrm{T}\parr)$$

$$\frac{P \vdash \Delta, y{:}A; \Theta \quad Q \vdash \Delta', x{:}B; \Theta}{(\nu y)x\langle y\rangle.(P \mid Q) \vdash \Delta, \Delta', x{:}A \otimes B; \Theta}\ (\mathrm{T}\otimes)$$

$$\frac{P \vdash \Delta, x{:}\overline{A}; \Theta \quad Q \vdash \Delta', x{:}A; \Theta}{(\nu x)(P \mid Q) \vdash \Delta, \Delta'; \Theta}\ (\mathsf{Tcut}) \qquad \frac{P \vdash y{:}\overline{A}; \Theta \quad Q \vdash \Delta; u{:}A, \Theta}{(\nu u)(!u(y).P \mid Q) \vdash \Delta; \Theta}\ (\mathsf{Tcut}^{?})$$

$$\frac{P \vdash \Delta, y{:}A; u{:}A, \Theta}{(\nu y)u\langle y\rangle.P \vdash \Delta; u{:}A, \Theta}\ (\mathsf{Tcopy})$$

$$\frac{P \vdash \Delta; u{:}A, \Theta}{P\{x/u\} \vdash \Delta, x{:}?A; \Theta}\ (\mathrm{T}?) \qquad \frac{Q \vdash y{:}A; \Theta}{!x(y).Q \vdash x{:}!A; \Theta}\ (\mathrm{T}!)$$

$$\frac{P \vdash \Delta, x{:}A; \Theta}{x.\mathtt{inl}; P \vdash \Delta, x{:}A \oplus B; \Theta}\ (\mathrm{T}\oplus_1) \qquad \frac{P \vdash \Delta, x{:}B; \Theta}{\vdash x.\mathtt{inr}; P \vdash \Delta, x{:}A \oplus B; \Theta}\ (\mathrm{T}\oplus_2)$$

$$\frac{P \vdash \Delta, x{:}A; \Theta \quad Q \vdash \Delta, x{:}B; \Theta}{x.\mathsf{case}(P, Q) \vdash \Delta, x{:}A \mathbin{\&} B; \Theta}\ (\mathrm{T}\&)$$

Fig. 8. The type system πCLL.

endpoints. Alternatively, we could postulate $\overline{\mathbf{1}} = \mathbf{1}$ (cf. $\overline{\mathsf{end}} = \mathsf{end}$) which would lead to a slight deviation from CLL and validate the nullary version of the mix rule (Girard 1987), making $\vdash \cdot$ derivable. This, and a further discussion of the general mix rule, are beyond the scope of this paper.

C-types may be assigned to π-calculus processes by following the key ideas of our interpretation, by a type system πCLL that corresponds this time (exactly) to Andreoli's dyadic system (Andreoli 1992). We present πCLL in Figure 8.

Typing judgments in πCLL have the (one sided) form $P \vdash \Delta; \Theta$, where P is a π-calculus process, Δ is a linear context, and Θ is a context with declarations $u{:}A$ which may be read as $u{:}?A$ in the original one-sided formulation of CLL. A remarkable consequence of the classical discipline is the loss of the *locality property* (Merro and Sangiorgi 2004) on shared channels, considered in general the most sensible for distributed implementations of shared channels. In the π-calculus, locality enforces that processes may not receive messages on previously received channel names, that is, only the output capability of a communication channel may be delegated in a distributed communication. In a very informal sense, non-locality means that a receiver may create a local stub for receiving a message on a shared channel created elsewhere, possibly causing undesirable interference. In πCLL, a well-typed process may input a channel of type $?A$, as the type $?A \multimap B$ now becomes expressible, thus breaking locality on shared channels, a behaviour excluded by the intuitionistic discipline.

We give an example of such a process. Let

$$C \triangleq (\nu x)(x(y).!y(z).P_x \mid (\nu q)x\langle q\rangle.(Q_q \mid R_x))$$

where, $Q_q \triangleq ((\nu k_1)q\langle k_1\rangle.Q_1 \mid (\nu k_2)q\langle k_2\rangle.Q_2)$. Notice that C contains the subsystem

$$S_x \triangleq x(y).!y(z).P_x.$$

The process S_x cannot be typed in πDILL. We may interpret S_x as a generator of remotely located shared server. S_x starts off by receiving a fresh channel name n from the sender client (on x) and then instantiates a replicated (receiver) shared server of the form $!n(z).P_n$. We illustrate the type derivation for subprocess S_x

1.	$P_v \vdash z{:}A; v{:}G$	(assumption, for some P_v)
2.	$!y(z).P_v \vdash y{:}!A; v{:}G$	(T! from 1)
3.	$!y(z).P_x \vdash y{:}!A, x{:}?G; \cdot$	(T? from 2)
4.	$x(y).!y(z).P_x \vdash x{:}!A \parr ?G$	(T$\parr$ from 3, notice that $!A \parr ?G = ?\overline{A} \multimap ?G$)

On the other hand, the subprocess $(\nu q)x\langle q\rangle.(Q_q \mid R_x)$ sends on x a fresh channel of type $?\overline{A}$, in which two (in this case) different requests will be issued to the remotely allocated server (see the definition of Q_q above).

Notice that most standard session type systems for the π-calculus, such as Gay and Hole (2005), also do not enforce locality of shared channels, so this feature of πCLL is not to be seen as a deviation from traditional session type systems. However, this non-local behaviour of shared channel names is not expressible in πDILL; we consider quite remarkable how moving from a classical to an intuitionistic session typing discipline (from a logical point of view), enforces the locality property, a behavioural concept related to high level properties of distributed communicating systems.

6. Further discussion

We further compare our linear type system for (finite) session types with more familiar session type systems (Kobayashi *et al.* 1996; Honda *et al.* 1998; Gay and Hole 2005). Arguably, apart from subtyping issues, which are out of the scope of this work, our type system is closely related to the one in Gay and Hole (2005), which presented the first session type system for the pure π-calculus.

An immediate observation is that in our case types are freely generated, while traditionally there is a stratification of types in 'session' and 'standard types' (the later corresponding to our $!A$ types, typing session initiation channels). In our interpretation, a session may either terminate ($\mathbf{1}$), or become a replicated server ($!A$), which reveals a more general and uniform type structure than the ones proposed in Gay and Hole (2005), Honda *et al.* (1998) and Kobayashi *et al.* (1996) which cannot express a type such as $A \multimap !S$, which describes a process than inputs a session of type A and then behaves as a replicated server of type $!S$. The possibility of finalizing a linear session with a replicated behaviour was also considered in Giunti and Vasconcelos (2010), as an addition to standard session types. In our setting, this arises naturally by accepting what the linear type structure offers us, for free.

Channel 'polarities' are captured in our system by the left-right distinction of sequents, rather than by annotations on channels (cf. x^+, x^-). Session and linear type systems (Kobayashi *et al.* 1996; Honda *et al.* 1998; Gay and Hole 2005) also include a typing rule for output of the form

$$\frac{\Gamma;\Delta \vdash P :: x{:}U}{\Gamma;\Delta, y{:}S \vdash x\langle y\rangle.P :: x{:}S\,!.U}\ \text{(T-Out)}$$

which in our system would correspond (by analogy) to

$$\frac{\Gamma;\Delta \vdash P :: x{:}C}{\Gamma;\Delta, y{:}A \vdash x\langle y\rangle.P :: x{:}A \otimes C}$$

In our case, an analogous rule may be derived by ⊗R and the copycat construction, where a 'proxy' for the free name *y*, bidirectionally copying behaviour *A*, is linked to *z*.

$$\frac{\Gamma;\Delta \vdash P :: x{:}C}{\Gamma;\Delta, y{:}A \vdash (\nu z)x\langle z\rangle.(id_A(y,z) \mid P) :: x{:}A \otimes C}.$$

The copycat $id_A(y,z)$ plays the role of the 'link' processes of Boreale (1998) and Sangiorgi (1996). Notice that in our case the definition of the 'link' is obtained for free by the interpretation of identity axioms (Proposition 3.1). The two processes depicted above can be shown to be behaviourally equivalent, under an adequate notion of typed observational equivalence, along the lines of Boreale (1998).

Concerning parallel composition of processes, usually two rules can be found, one rule corresponding to the interconnection of two dual session endpoints (implemented by a name restriction rule), and other rule corresponding to independent parallel composition, also present in most linear type systems for mobile processes. In our case, the cut rule combines both principles, and the following rule is derivable:

$$\frac{\Gamma;\Delta \vdash P :: -{:}\mathbf{1} \quad \Gamma;\Delta' \vdash Q :: T}{\Gamma;\Delta,\Delta' \vdash P \mid Q :: T}\ \text{(comp)}.$$

A consequence of the logical nature of our composition principles cut and $\text{cut}^!$ is that our typing discipline intrinsically enforces global progress, unlike with traditional session type systems (Honda *et al.* 1998; Gay and Hole 2005), which do not ensure progress in general, as we achieve in this work. The session composition rule found in such systems does not take into account the causal dependency between different sessions, and validates the connection of two session endpoints just by requiring duality of their polarity. For example, the system of Gay and Hole (2005) contains a type rule of the form

$$\frac{\Gamma, x^+{:}S, x^-{:}S' \vdash P \quad S \perp S'}{\Gamma \vdash (\boldsymbol{\nu} x{:}S)P}$$

where $S \perp S'$ expresses the duality of the session types S and S'. This rule ensures that only dual protocols are connected to form a session, but of course does not prevent causality cycles from appearing in systems with multiple open sessions. Techniques to ensure progress in traditional session type systems, typically building on extraneous devices such as well-founded orderings on events, have been proposed by several authors (Kobayashi 1998; Dezani-Ciancaglini *et al.* 2008). We note that our formulation of progress is similar to that of Carbone and Debois (2010), which uses reduction contexts and live channels, making the reasoning more complex than in our formulation.

A minor difference of our system and those typically presented in the literature is the inclusion of recursive types, which is largely orthogonal to our development. Adding recursive types to our system is straightforward, but similarly to what happens in λ-calculi the connections with logic disappear. The study of inductive and co-inductive session types can re-establish this connection and is a goal of future work.

We find it important to identify systems of session types such as the one we have identified, in which progress (deadlock-freedom) is an essential meta-theoretical property, as is the case of basic type systems for foundational programming languages, in particular, for typed λ-calculi. Moreover, we have not been able to find an example of an interesting session typed system not typeable in our framework. A simple example of a system typeable in Gay and Hole (2005) is

$$\Gamma \vdash (\boldsymbol{\nu} x)(\boldsymbol{\nu} y)(x\langle z\rangle.y\langle s\rangle.\mathbf{0} \mid x(w).y(v).\mathbf{0}).$$

This process progresses by the coordination of two different sessions, one on x and another on y, terminating (reducing to $\mathbf{0}$) in two communication steps. Likewise, the type system in Gay and Hole (2005) also types the process

$$\Gamma \vdash (\boldsymbol{\nu} x)(\boldsymbol{\nu} y)(x\langle z\rangle.y\langle s\rangle.\mathbf{0} \mid y(v).x(w).\mathbf{0})$$

which is stuck (standard type systems for session types do not satisfy general progress). Our type system does not type any of these two examples, as the global progress property it enforces relies on forbidding certain forms of inter-session causal dependence, that the first example above fails to comply with. Essentially, note that sessions x and y are globally coordinated as if they were the same single session n_{xy}, cf. the following (typeable in πDILL) process:

$$\Gamma \vdash (\boldsymbol{\nu} n_{xy})(n_{xy}\langle z\rangle.n_{xy}\langle s\rangle.\mathbf{0} \mid n_{xy}(w).n_{xy}(v).\mathbf{0}).$$

Processes typeable in our system satisfy such inter-session causal independence property, which in turn enforces global progress, as stated in Theorem 4.5. In Pérez *et al.* (2012), we discuss these interesting issues in the context of typed observational equivalences for our session typed language.

The work reported in this paper started a research program in which structured concurrency communication is approached from several perspectives, always based on canonical logical principles. For example, in Toninho *et al.* (2011) we have shown how to specify rich interface contracts with dependent types, while in Pfenning *et al.* (2011) we have introduced proof-carrying code with digital certificates in our basic framework, always based on purely logical constructions.

There are not many studies comparing the expressiveness of session type systems, and that also clearly seems a challenging research direction. An important instrument towards that goal is, we believe, a better understanding of observational equivalences under the session type discipline.

7. Related work and conclusion

We have established a tight correspondence between a session-based type discipline for the π-calculus and intuitionistic linear logic: typing rules correspond to dual intuitionistic linear sequent calculus proof rules, moreover process reduction may be simulated in a type preserving way by proof conversions and reductions, and *vice versa*. As a result, we obtain the subject reduction property, from which session fidelity follows. Our basic typing discipline intrinsically ensures global progress, beyond the restricted 'progress on a single session' property obtained in pure session type systems.

Other works have investigated π-calculus models of linear logic proofs. Bellin and Scott (1994) establish a mapping from linear logic proofs to a variant of the π-calculus and some connections between proof reduction and π-calculus reduction. However, this mapping results in complex encodings, so that their system could hardly be considered a type assignment system for processes, which has been achieved in this work. Moreover, no relation between behavioural descriptions and logical propositions was identified, as put by the authors: '[our encodings] have less to do with logic than one might think, they are essentially only about the abstract pluggings in proof structures'.

A realizability interpretation for a linear logic augmented with temporal modalities (cf. Hennessy–Milner) was proposed in Beffara (2006), also based on a π-calculus variant. A recent related development is Honda and Laurent (2010), where a correspondence between (independently formulated) proof nets and an IO-typed π-calculus is established. In our case, the type system and the logic proof system are exactly the same, and we reveal a direct connection between pure linear logic propositions and behavioural types on π-calculus, that covers all (both shared and linear) features of finite session types. A development of session types as linear process types (in the sense of Kobayashi *et al.* (1996)) is presented in Giunti and Vasconcelos (2010), where linearity and sharing are expressed by special annotations, unrelated to a linear logic interpretation.

We have also analysed the relation between our type discipline and (finite, deadlock-free) session types. It is important to notice that our interpretation does not require locality (Merro and Sangiorgi 2004) for linear session channels, under which only the output capability of names could be transmitted, which seems required in other works on linearity for π-calculi (e.g. Yoshida *et al.* (2007)). On the other hand, our intuitionistic discipline enforces locality of shared channels, which, quite interestingly, seems to be the sensible choice for distributed implementations of sessions. Further related topics would be the accommodation of recursive types and logical relations (Caires 2007).

One important motivation for choosing a purely logical approach to typing is that it often suggests uniform and expressive generalizations. In ongoing work, we have also established an explicit relationship between session-based concurrency and typed functional computation where in both cases determinacy (no races) and progress (deadlock-freedom) are expected properties. In particular, we have investigated new encodings of λ-calculi into the π-calculus that arise from translations from DILL natural deduction into sequent calculus and the reduction strategies they induce (Toninho *et al.* 2012). We have also explored a dependent generalization of our system of simple linear types, which successfully captures many additional properties of communication behaviour in a purely logical manner (Toninho *et al.* 2011). Furthermore, the combination of dependent session types, proof irrelevance and a notion of affirmation allows us to capture a high-level model of certified, distributed code (Pfenning *et al.* 2011). In Caires *et al.* (2012) we have extended our interpretation to second-order intuitionistic linear logic, bringing session polymorphism within the scope of our work. We have also studied reasoning techniques based on relational parametricity. Building on Caires and Pfenning (2010) and Wadler (2012) develops an interpretation of second-order CLL as a polymorphic session calculus. Wadler's system is substantially more distant from a typical process calculus since it admits prefix commutations and reduction under prefixes as primitive reductions, while we map commuting conversions to structural congruence and observational equivalence, and computational conversions to reduction. Moreover, his presentation of linear logic follows more closely the original one of Girard, using explicit rules for weakening, contraction and dereliction instead of a copy rule and the dual formulation. Thus, server replication is internalized through a cut of the ! rule and the contraction rule. Since contraction is mapped to a meta-level renaming in the style of our $T?$ rule, the operational interpretation of spawning a server becomes a bit unnatural from a process calculus perspective.

Acknowledgments

We thank Vasco T. Vasconcelos, Simon Gay, Nobuko Yoshida, Kohei Honda and the anonymous referees for several useful comments. Support for this research was provided by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the Carnegie Mellon Portugal Program, under grants SFRH/BD/33763/2009, INTERFACES NGN-44/2009, CITI strategic project PEst-OE/EEI/UI0527/2011 and by the Army Research Office under Award No. W911NF-09-1-0273.

Appendix A. Proofs

A.1. *Inversion lemmas*

Lemma A.1. Let $\Gamma; \Delta \vdash D \rightsquigarrow P :: x : C$.

1. If $P \xrightarrow{\alpha} Q$ and $C = \mathbf{1}$ then $s(\alpha) \neq x$.
2. If $P \xrightarrow{\alpha} Q$ and $y : \mathbf{1} \in \Delta$ then $s(\alpha) \neq y$.
3. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \otimes B$ then $\alpha = \overline{(\nu y)x\langle y\rangle}$.
4. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \otimes B \in \Delta$ then $\alpha = y(z)$.
5. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \multimap B$ then $\alpha = x(y)$.
6. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \multimap B \in \Delta$ then $\alpha = \overline{(\nu z)y\langle z\rangle}$.
7. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \,\&\, B$ then $\alpha = x.\mathsf{inl}$ or $\alpha = x.\mathsf{inr}$.
8. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \,\&\, B \in \Delta$ then $\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$.
9. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \oplus B$ then $\alpha = \overline{x.\mathsf{inl}}$ or $\alpha = \overline{x.\mathsf{inr}}$.
10. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \oplus B \in \Delta$ then $\alpha = y.\mathsf{inl}$ or $\alpha = y.\mathsf{inr}$.
11. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = !A$ then $\alpha = x(y)$.

12. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : !A$ or $y \in \Gamma$ then $\alpha = \overline{(\nu z)y\langle z\rangle}$.

Proof. By induction on the structure of D.

1. If $P \xrightarrow{\alpha} Q$ and $C = \mathbf{1}$ then $s(\alpha) \neq x$.

Case:copy, all left rules except $\mathbf{1}$L and !L

$s(\alpha) \neq x$ by the definition of the l.t.s.

Case:$\mathbf{1}$L or !L

$s(\alpha) \neq x$ by i.h.

Case:cut D_1 $(y. D_2)$

$D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $P_1 \xrightarrow{\alpha} Q_1$

$s(\alpha) \neq x$ trivial, since $x \notin fn(P_1)$ by typing

Subcase: $P_2 \xrightarrow{\alpha} Q_2$

$s(\alpha) \neq x$ by i.h. on D_2

Case:cut$^!$ D_1 $(u. D_2)$

$D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $P_1 \xrightarrow{\alpha} Q_1$

$s(\alpha) \neq x$ trivial, since $x \notin fn(P_1)$ by typing

Subcase: $P_2 \xrightarrow{\alpha} Q_2$

$s(\alpha) \neq x$ by i.h. on D_2

Case:all other rules do not offer $\xrightarrow{\alpha}$ or $C \neq \mathbf{1}$

2. If $P \xrightarrow{\alpha} Q$ and $y : \mathbf{1} \in \Delta$ then $s(\alpha) \neq y$.

Case:copy, all left rules except $\mathbf{1}$L and !L

$s(\alpha) \neq y$ by the definition of the l.t.s.

Case:$\mathbf{1}$L or !L

$s(\alpha) \neq y$ by i.h.

Case:$\Gamma; \Delta_1, \Delta_2 \vdash$ cut D_1 $(z. D_2) \rightsquigarrow P :: x : C$, with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $y : \mathbf{1} \in \Delta_1$

$s(\alpha) \neq y$ by i.h. and $y \notin fn(P_2)$

Subcase: $y : \mathbf{1} \in \Delta_2$ with $z \neq y$

$s(\alpha) \neq y$ by i.h. and $y \notin fn(P_1)$

Case:cut$^!$ D_1 $(u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $P_1 \xrightarrow{\alpha} Q_1$

$s(\alpha) \neq y$ trivial, since linear ctxt. is empty for D_1 and $y : \mathbf{1}$

Subcase: $P_2 \xrightarrow{\alpha} Q_2$

$s(\alpha) \neq y$ by i.h.

3. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \otimes B$ then $\alpha = \overline{(\nu y)x\langle y\rangle}$.

   Case:$\mathsf{1L}$ or $\mathsf{!L}$

   $\alpha = \overline{(\nu y)x\langle y\rangle}$ by i.h.

   Case:$\otimes\mathsf{R}$

   $\alpha = \overline{(\nu y)x\langle y\rangle}$ by the l.t.s

   Case:$\mathsf{cut}\ D_1\ (y.D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

   $\alpha = \overline{(\nu y)x\langle y\rangle}$ by i.h. on D_2 and $x \notin fn(P_1)$

   Case:$\mathsf{cut}^!\ D_1\ (u.D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

   $\alpha = \overline{(\nu y)x\langle y\rangle}$ by i.h. on D_2

   Case: All other rules do not have $s(\alpha) = x$ and $C = A \otimes B$

4. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \otimes B \in \Delta$ then $\alpha = y(z)$.

   Case:$\mathsf{1L}$ or $\mathsf{!L}$

   $\alpha = y(z)$ by i.h.

   Case:$\otimes\mathsf{L}$

   $\alpha = y(z)$ by the l.t.s

   Case:$\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{cut}\ D_1\ (z.D_2) \rightsquigarrow P :: x : C$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

   Subcase: $y \in \Delta_1$
   $\alpha = y(z)$ by i.h. on D_1 and $y \notin fn(P_2)$

   Subcase: $y \in \Delta_2$
   $\alpha = y(z)$ by i.h. on D_2 and $y \notin fn(P_1)$

   Case:$\mathsf{cut}^!\ D_1\ (u.D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

   $\alpha = y(z)$ by i.h. on D_2

5. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \multimap B$ then $\alpha = x(y)$.

   Case:$\mathsf{1L}$ or $\mathsf{!L}$

   $\alpha = x(y)$ by i.h.

   Case:$\multimap\mathsf{R}$

   $\alpha = x(y)$ by the l.t.s

   Case:$\mathsf{cut}\ D_1\ (y.D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

   $\alpha = x(y)$ by i.h. on D_2 and $x \notin fn(P_1)$

   Case:$\mathsf{cut}^!\ D_1\ (u.D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

   $\alpha = x(y)$ by i.h. on D_2

   Case: All other rules do not have $s(\alpha) = x$ and $C = A \multimap B$

6. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \multimap B \in \Delta$ then $\alpha = \overline{(\nu z)y\langle z\rangle}$.

   Case:$\mathsf{1L}$ or $\mathsf{!L}$

   $\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h.

Case:$\multimap$L

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by the l.t.s

Case:$\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{cut}\ D_1\ (w. D_2) \rightsquigarrow P :: x : C$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $y \in \Delta_1$

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h. on D_1 and $y \notin fn(P_2)$

Subcase: $y \in \Delta_2$

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h. on D_2 and $y \notin fn(P_1)$

Case:$\mathsf{cut}^!\ D_1\ (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h. on D_2

7. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \,\&\, B$ then $\alpha = x.\mathsf{inl}$ or $\alpha = x.\mathsf{inr}$.

Case:$\mathbf{1}$L or !L

$\alpha = x.\mathsf{inl}$ or $\alpha = x.\mathsf{inr}$ by i.h.

Case:$\&$R

$\alpha = x.\mathsf{inl}$ or $\alpha = x.\mathsf{inr}$ by the l.t.s

Case:$\mathsf{cut}\ D_1\ (y. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = x(y)$ by i.h. on D_2 and $x \notin fn(P_1)$

Case:$\mathsf{cut}^!\ D_1\ (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = x(y)$ by i.h. on D_2

Case: All other rules do not have $s(\alpha) = x$ and $C = A \,\&\, B$

8. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \,\&\, B \in \Delta$ then $\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$.

Case:$\mathbf{1}$L or !L

$\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$ by i.h.

Case:$\&\mathsf{L}_1$

$\alpha = \overline{y.\mathsf{inl}}$ by the l.t.s

Case:$\&\mathsf{L}_2$

$\alpha = \overline{y.\mathsf{inr}}$ by the l.t.s

Case:$\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{cut}\ D_1\ (w. D_2) \rightsquigarrow P :: x : C$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $y \in \Delta_1$

$\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$ by i.h. on D_1 and $y \notin fn(P_2)$

Subcase: $y \in \Delta_2$

$\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$ by i.h. on D_2 and $y \notin fn(P_1)$

Case:$\mathsf{cut}^!\ D_1\ (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$ by i.h. on D_2

9. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \oplus B$ then $\alpha = \overline{x.\mathsf{inl}}$ or $\alpha = \overline{x.\mathsf{inr}}$.

Case:$\mathbf{1}$L or !L

$\alpha = \overline{x.\mathsf{inl}}$ or $\alpha = \overline{x.\mathsf{inr}}$ by i.h.

Case:$\oplus\mathsf{R}_1$

$\alpha = \overline{x.\mathsf{inl}}$ by the l.t.s

Case:$\oplus\mathsf{R}_2$

$\alpha = \overline{x.\mathsf{inr}}$ by the l.t.s

Case:$\mathsf{cut}\ D_1\ (y.\,D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = \overline{x.\mathsf{inl}}$ or $\alpha = \overline{x.\mathsf{inr}}$ by i.h. on D_2 and $x \notin fn(P_1)$

Case:$\mathsf{cut}^!\ D_1\ (u.\,D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = \overline{x.\mathsf{inl}}$ or $\alpha = \overline{x.\mathsf{inr}}$ by i.h. on D_2

Case: All other rules do not have $s(\alpha) = x$ and $C = A \oplus B$

10. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \oplus B \in \Delta$ then $\alpha = y.\mathsf{inl}$ or $\alpha = y.\mathsf{inr}$.

Case:$\mathbf{1}\mathsf{L}$ or $!\mathsf{L}$

$\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$ by i.h.

Case:$\oplus\mathsf{L}$

$\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$ by the l.t.s

Case:$\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{cut}\ D_1\ (w.\,D_2) \rightsquigarrow P :: x : C$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $y \in \Delta_1$

$\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$ by i.h. on D_1 and $y \notin fn(P_2)$

Subcase: $y \in \Delta_2$

$\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$ by i.h. on D_2 and $y \notin fn(P_1)$

Case:$\mathsf{cut}^!\ D_1\ (u.\,D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = \overline{y.\mathsf{inl}}$ or $\alpha = \overline{y.\mathsf{inr}}$ by i.h. on D_2

11. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = !A$ then $\alpha = x(y)$.

Case:$\mathbf{1}\mathsf{L}$ or $!\mathsf{L}$

$\alpha = x(y)$ by i.h.

Case:$!\mathsf{R}$

$\alpha = x(y)$ by the l.t.s

Case:$\mathsf{cut}\ D_1\ (z.\,D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = x(y)$ by i.h. on D_2 and $x \notin fn(P_1)$

Case:$\mathsf{cut}^!\ D_1\ (u.\,D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = x(y)$ by i.h. on D_2

Case: All other rules do not have $s(\alpha) = x$ and $C = !A$

12. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : !A$ or $y \in \Gamma$ then $\alpha = \overline{(\nu z)y\langle z\rangle}$.

Case:$\mathbf{1}\mathsf{L}$ or $!\mathsf{L}$

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h.

Case:copy

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by the l.t.s

Case:$\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{cut}\ D_1\ (w. D_2) \rightsquigarrow P\ ::\ x : C$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $y : !A$ and $y \in \Delta_1$

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h. on D_1 and $y \notin fn(P_2)$

Subcase: $y : !A$ and $y \in \Delta_2$

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h. on D_2 and $y \notin fn(P_1)$

Subcase: $y \in \Gamma$ and $P \xrightarrow{\alpha} Q$ from $P_1 \xrightarrow{\alpha} Q_1$

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h. on D_1

Subcase: $y \in \Gamma$ and $P \xrightarrow{\alpha} Q$ from $P_2 \xrightarrow{\alpha} Q_2$

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h. on D_2

Case:$\mathsf{cut}^!\ D_1\ (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $y : !A$

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h. on D_2

Subcase: $y \in \Gamma$

$\alpha = \overline{(\nu z)y\langle z\rangle}$ by i.h. on D_2

□

A.2. *Preservation lemmas*

Lemma A.2. Assume

a. $\Gamma; \Delta_1 \vdash D \rightsquigarrow P\ ::\ x{:}C_1 \mathbin{\&} C_2$ with $P \xrightarrow{x.\mathsf{inl}} P'$;

b. $\Gamma; \Delta_2, x{:}C_1 \mathbin{\&} C_2 \vdash E \rightsquigarrow Q\ ::\ z{:}C$ with $Q \xrightarrow{\overline{x.\mathsf{inl}}} Q'$.

Then,

c. $\mathsf{cut}\ D(x. E) \equiv\Rightarrow\equiv F$ for some F;

d. $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R\ ::\ z : C$ for some $R \equiv (\nu x)(P' \mid Q')$.

Proof. By simultaneous induction on D and E. That is, in each appeal to the induction hypothesis either D becomes smaller and E remains the same, or D remains the same and E becomes smaller.

The possible cases for D are $\mathsf{\&R}$, $\mathbf{1}\mathsf{L}$, $!\mathsf{L}$, cut and $\mathsf{cut}^!$. In all other cases P^x cannot offer $x.\mathsf{inl}$. The possible cases for E are $\mathsf{\&L}_1$, $\mathbf{1}\mathsf{L}$, $!\mathsf{L}$, cut and $\mathsf{cut}^!$.

Case:$D = \mathsf{\&R}\ D_1\ D_2$ and $E = \mathsf{\&L}_1\ x\ (x. E_1)$.

$\mathsf{cut}\ D\ (x. E) = \mathsf{cut}\ (\mathsf{\&R}\ D_1\ D_2)\ (\mathsf{\&L}_1\ x\ (x. E_1)) = F$ [a]

$F \rightsquigarrow (\nu x)(x.\mathsf{case}(P_1^x, P_2^x) \mid x.\mathsf{inl}; Q_1^z) = (\nu x)(P^x \mid Q^z)$ [b]

$F \Rightarrow \mathsf{cut}\ D_1\ (x. E_1) = F'$ by reduction $(\mathsf{cut}/\mathsf{\&R}/\mathsf{\&L}_1)$ [c]

$F' \rightsquigarrow (\nu x)(P_1^x \mid Q_1^z) \equiv R'$ with $P_1^x = P'^x$ and $Q_1^z = Q'^z$. [d]

Case:$D = \mathbf{1}\mathsf{L}\ y\ D_1$ and E arbitrary.

$\mathsf{cut}\ D\ (x.E) = \mathsf{cut}\ (\mathbf{1}\mathsf{L}\ y\ D_1)\ (x.E)$
$\equiv \mathbf{1}\mathsf{L}\ y\ (\mathsf{cut}\ D_1\ (x.E)) = \mathrm{F}$ by rule (cut/$\mathbf{1}$L/—) [a]

$F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R$ since $D_1 \rightsquigarrow P^x$ [b]

$\mathsf{cut}\ D_1\ (x.E) \equiv F_1$ for some F_1;
$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z)$;
$F_1 \Rightarrow F_1'$;
$F_1' \rightsquigarrow R_1'^z \equiv (\nu x)(P'^x \mid Q'^z)$ by i.h. on D_1, E

$F = \mathbf{1}\mathsf{L}\ y\ F_1 \Rightarrow \mathbf{1}\mathsf{L}\ y\ F_1' = F'$ by congruence [c]
$F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$D = \,!\mathsf{L}\ y\ (u.D_1)$ and E arbitrary.

$\mathsf{cut}\ D\ (x.E) = \mathsf{cut}\ (!\mathsf{L}\ y\ (u.D_1))\ (x.E)$
$\equiv \,!\mathsf{L}\ y\ (u.\mathsf{cut}\ D_1\ (x.E)) = F$ by rule (cut/ !L/—) [a]

$F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R$ since $D_1 \rightsquigarrow P^x\{u/y\}$ [b]

$\mathsf{cut}\ D_1\ (x.E) \equiv F_1$ for some F_1;
$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z)$;
$F_1 \Rightarrow F_1'$;
$F_1' \rightsquigarrow R_1'^z \equiv (\nu x)(P'^x \mid Q'^z)$ by i.h. on D_1, E

$F = \,!\mathsf{L}\ y\ (u.F_1) \Rightarrow \,!\mathsf{L}\ y\ (u.F_1') = F'$ by congruence [c]
$F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$D = \mathsf{cut}\ D_1\ (y.D_2)$ and E arbitrary.

$\mathsf{cut}\ D\ (x.E) = \mathsf{cut}\ (\mathsf{cut}\ D_1\ (y.D_2))\ (x.E)$
$\equiv \mathsf{cut}\ D_1\ (y.\mathsf{cut}\ D_2\ (x.E)) = F$ by rule (cut/—/cut$_1$) [a]

$F \rightsquigarrow (\nu y)(P_1^y \mid ((\nu x)(P_2^x \mid Q^z))) \equiv (\nu x)((\nu y)(P_1^y \mid P_2^x) \mid Q^z) = R$ [b]

$\mathsf{cut}\ D_2\ (x.E) \equiv F_2$ with
$F_2 \rightsquigarrow R_2$ and $R_2 \equiv (\nu x)(P_2^x \mid Q^z)$ and
$F_2 \Rightarrow F_2'$ with
$F_2' \rightsquigarrow R_2' \equiv (\nu x)(P_2'^x \mid Q'^z)$ by i.h. on D_2, E

$F \Rightarrow \mathsf{cut}\ D_1\ (y.F_2') = F'$ by congruence [c]
$F' \rightsquigarrow (\nu y)(P_1^y \mid (\nu x)(P_2'^x \mid Q'^z)) \equiv (\nu x)((\nu y)(P_1^y \mid P_2'^x) \mid Q'^z)$
$= (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$D = \mathsf{cut}^!\ D_1\ (u.D_2)$ and E arbitrary.

$\mathsf{cut}\ D\ (x.E) = \mathsf{cut}\ (\mathsf{cut}^!\ D_1\ (u.D_2))\ (x.E)$
$\equiv \mathsf{cut}^!\ D_1\ (u.\mathsf{cut}\ D_2\ (x.E)) = F$ by rule (cut/cut$^!$/—) [a]

$F \rightsquigarrow (\nu u)((!u(y).P_1^y) \mid (\nu x)(P_2^x \mid Q^z))$
$\equiv (\nu x)((\nu u)((!u(y).P_1^y) \mid P_2^x) \mid Q^z) = R$ by struct. cong. [b]

$\mathsf{cut}\ D_2\ (x.E) \equiv F_2$ with
$F_2 \rightsquigarrow R_2$ and $R_2 \equiv (\nu x)(P_2^x \mid Q^z)$ and
$F_2 \Rightarrow F_2'$ with
$F_2' \rightsquigarrow R_2' \equiv (\nu x)(P_2'^x \mid Q'^z)$ by i.h. on D_2, E

$F \Rightarrow \mathsf{cut}^! \; D_1 \; (u. F_2') = F'$ by congruence [c]

$F' \rightsquigarrow (\nu u)((!u(y).P_1^y) \mid (\nu x)(P_2'^x \mid Q'^z))$
$\equiv (\nu x)((\nu u)((!u(y).P_1^y) \mid P_2'^x) \mid Q'^z) = (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$E = \mathbf{1}\mathsf{L} \; y \; E_1$ and D arbitrary.

$\mathsf{cut} \; D \; (x. E) = \mathsf{cut} \; D \; (x. \mathbf{1}\mathsf{L} \; y \; E_1)$
$\equiv \mathbf{1}\mathsf{L} \; y \; (\mathsf{cut} \; D \; (x. E_1)) = F$ by rule $(\mathsf{cut}/-/\mathbf{1}\mathsf{L})$ [a]

$F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R$ [b]

$\mathsf{cut} \; D \; (x. E_1) \equiv F_1$ for some F_1;
$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z)$;
$F_1 \Rightarrow F_1'$;
$F_1' \rightsquigarrow R_1' \equiv (\nu x)(P'^x \mid Q'^z)$ by i.h. on D, E_1

$F = \mathbf{1}\mathsf{L} \; y \; F_1 \Rightarrow \mathbf{1}\mathsf{L} \; y \; F_1' = F'$ by congruence [c]

$F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$E = !\mathsf{L} \; y \; (u. E_1)$ and D arbitrary.

$\mathsf{cut} \; D \; (x. E) = \mathsf{cut} \; D \; (x. !\mathsf{L} \; y \; (u. E_1))$
$\equiv !\mathsf{L} \; y \; (u. \mathsf{cut} \; D \; (x. E_1)) = F$ by rule $(\mathsf{cut}/-/!\mathsf{L})$ [a]

$F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R$ since $E_1 \rightsquigarrow Q^z\{u/y\}$ [b]

$\mathsf{cut} \; D \; (x. E_1) \equiv F_1$ for some F_1;
$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z)$;
$F_1 \Rightarrow F_1'$;
$F_1' \rightsquigarrow R_1' \equiv (\nu x)(P'^x \mid Q'^z)$ by i.h. on D, E_1

$F = !\mathsf{L} \; y \; (u. F_1) \Rightarrow \mathbf{1}\mathsf{L} \; y \; (u. F_1') = F'$ by congruence [c]

$F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$E = \mathsf{cut} \; E_1 \; (y. E_2)$ with $x \in FV(E_1)$ and D arbitrary.

$\mathsf{cut} \; D \; (x. E) = \mathsf{cut} \; D \; (x. \mathsf{cut} \; E_1 \; (y. E_2))$
$\equiv \mathsf{cut} \; (\mathsf{cut} \; D \; (x. E_1)) \; (y. E_2) = F$ by reduction $(\mathsf{cut}/-/\mathsf{cut}_1)$ [a]

$F \rightsquigarrow (\nu y)((\nu x)(P^x \mid Q_1^y) \mid Q_2^z) \equiv (\nu x)(P^x \mid (\nu y)(Q_1^y \mid Q_2^z)) = R$ [b]

$\mathsf{cut} \; D \; (x. E_1) \equiv F_1$ with
$F_1 \rightsquigarrow R_1 \equiv (\nu x)(P^x \mid Q_1^y)$ and
$F_1 \Rightarrow F_1'$ with
$F_1' \rightsquigarrow R_1' \equiv (\nu x)(P'^x \mid Q_1'^y)$ by i.h. on D, E_1

$F \Rightarrow \mathsf{cut} \; F_1' \; (y. E_2) = F'$ by congruence [c]
$F' \rightsquigarrow (\nu y)((\nu x)(P'^x \mid Q_1'^y) \mid Q_2^z) \equiv (\nu x)(P'^x \mid (\nu y)(Q_1'^y \mid Q_2^z))$
$= (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$E = \mathsf{cut} \; E_1 \; (y. E_2)$ with $x \in FV(E_2)$ and D arbitrary.

$\mathsf{cut} \; D \; (x. E) = \mathsf{cut} \; D \; (x. \mathsf{cut} \; E_1 \; (y. E_2))$
$\equiv \mathsf{cut} \; E_1 \; (y. \mathsf{cut} \; D \; (x. E_2)) = F$ by reduction $(\mathsf{cut}/-/\mathsf{cut}_2)$ [a]

$F \rightsquigarrow (\nu y)(Q_1^y \mid (\nu x)(P^x \mid Q_2^z)) \equiv (\nu x)(P^x \mid (\nu y)(Q_1^y \mid Q_2^z)) = R$ [b]

$\mathsf{cut}\ D\ (x.\,E_2) \equiv F_2$ with
$F_2 \rightsquigarrow R_2 \equiv (\nu x)(P^x \mid Q_2^z)$ and
$F_2 \Rightarrow F_2'$ with
$F_2' \rightsquigarrow R_2' \equiv (\nu x)(P'^x \mid Q_2'^z)$ by i.h. on D, E_2
$F \Rightarrow \mathsf{cut}\ E_1\ (y.\,F_2') = F'$ by congruence [c]
$F' \rightsquigarrow (\nu y)(Q_1^y \mid (\nu x)(P'^x \mid Q_2'^z)) \equiv (\nu x)(P'^x \mid (\nu y)(Q_1^y \mid Q_2'^z))$
$= (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$E = \mathsf{cut}^!\ E_1\ (u.\,E_2)$ with $x \in FV(E_1)$ and D arbitrary. This case is impossible because E_1 contains no free linear variables.

Case:$E = \mathsf{cut}^!\ E_1\ (u.\,E_2)$ with $x \in FV(E_2)$ and D arbitrary.

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\,\mathsf{cut}^!\ E_1\ (u.\,E_2))$
$\equiv \mathsf{cut}^!\ E_1\ (u.\,\mathsf{cut}\ D\ (x.\,E_2)) = F$ by reduction ($\mathsf{cut}/-/\mathsf{cut}^!$) [a]

$F \rightsquigarrow (\nu u)((!u(y).Q_1^y) \mid (\nu x)(P^x \mid Q_2^z)) \equiv$
$(\nu x)(P^x \mid (\nu u)((!u(y).Q_1^y) \mid Q_2^z)) = R$ [b]

$\mathsf{cut}\ D\ (x.\,E_2) \equiv F_2$ with
$F_2 \rightsquigarrow R_2 \equiv (\nu x)(P^x \mid Q_2^z)$ and
$F_2 \Rightarrow F_2'$ with
$F_2' \rightsquigarrow R_2' \equiv (\nu x)(P'^x \mid Q_2'^z)$ by i.h. on D, E_2
$F \Rightarrow \mathsf{cut}^!\ E_1\ (u.\,F_2') = F'$ by congruence [c]
$F' \rightsquigarrow (\nu u)((!u(y).Q_1^y) \mid (\nu x)(P'^x \mid Q_2'^z))$
$\equiv (\nu x)(P'^x \mid (\nu u)((!u(y).Q_1^y) \mid Q_2'^z)) = (\nu x)(P'^x \mid Q'^z)$ [d]

□

Lemma A.3. Assume

a. $\Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x{:}C_1 \mathbin{\&} C_2$ with $P \xrightarrow{x.\mathsf{inr}} P'$;
b. $\Gamma; \Delta_2, x{:}C_1 \mathbin{\&} C_2 \vdash E \rightsquigarrow Q :: z{:}C$ with $Q \xrightarrow{\overline{x.\mathsf{inr}}} Q'$.

Then

c. $\mathsf{cut}\ D(x.\,E) \equiv\Rightarrow\equiv F$ for some F;
d. $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu x)(P' \mid Q')$.

Proof. Completely symmetric to the previous lemma. □

Lemma A.4. Assume

a. $\Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x{:}C_1 \oplus C_2$ with $P \xrightarrow{\overline{x.\mathsf{inl}}} P'$;
b. $\Gamma; \Delta_2, x{:}C_1 \oplus C_2 \vdash E \rightsquigarrow Q :: z{:}C$ with $Q \xrightarrow{x.\mathsf{inl}} Q'$.

Then

c. $\mathsf{cut}\ D(x.\,E) \equiv\Rightarrow\equiv F$ for some F;
d. $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu x)(P' \mid Q')$.

Proof. By simultaneous induction on D and E. That is, in each appeal to the induction hypothesis either D becomes smaller and E remains the same, or D remains the same and E becomes smaller.

The possible cases for D are $\oplus\mathsf{R}_1$, $\mathbf{1}\mathsf{L}$, $!\mathsf{L}$, cut and $\mathsf{cut}^!$. In all other cases P^x cannot offer $\overline{x.\mathsf{inl}}$. The possible cases for E are $\oplus\mathsf{L}$, $\mathbf{1}\mathsf{L}$, $!\mathsf{L}$, cut and $\mathsf{cut}^!$.

Case:$D = \oplus\mathsf{R}_1\ D_1$ and $E = \oplus\mathsf{L}\ x\ (x.\,E_1)\ (x.\,E_2)$.

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (\oplus\mathsf{R}_1\ D_1)\ (\oplus\mathsf{L}\ x\ (x.\,E_1)\ (x.\,E_2)) = F$ [a]

$F \rightsquigarrow (\nu x)(x.\mathsf{inl}; P_1^x \mid x.\mathsf{case}(Q_1^z, Q_2^z)) = (\nu x)(P^x \mid Q^z)$ [b]

$F \Rightarrow \mathsf{cut}\ D_1\ (x.\,E_1) = F'$ by reduction $(\mathsf{cut}/\oplus\mathsf{R}_1/\oplus\mathsf{L})$ [c]

$F' \rightsquigarrow (\nu x)(P_1^x \mid Q_1^z) \equiv R'$ with $P_1^x = P'^x$ and $Q_1^z = Q'^z$ [d]

Case:$D = \mathbf{1}\mathsf{L}\ y\ D_1$ and E arbitrary.

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (\mathbf{1}\mathsf{L}\ y\ D_1)\ (x.\,E)$
$\equiv \mathbf{1}\mathsf{L}\ y\ (\mathsf{cut}\ D_1\ (x.\,E)) = \mathrm{F}$ by rule $(\mathsf{cut}/\mathbf{1}\mathsf{L}/-)$ [a]

$F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R$ since $D_1 \rightsquigarrow P^x$ [b]

$\mathsf{cut}\ D_1\ (x.\,E) \equiv F_1$ for some F_1;
$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z)$;
$F_1 \Rightarrow F_1'$;
$F_1' \rightsquigarrow R_1'^z \equiv (\nu x)(P'^x \mid Q'^z)$ by i.h. on D_1, E

$F = \mathbf{1}\mathsf{L}\ y\ F_1 \Rightarrow \mathbf{1}\mathsf{L}\ y\ F_1' = F'$ by congruence [c]
$F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$D = !\mathsf{L}\ y\ (u.\,D_1)$ and E arbitrary.

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (!\mathsf{L}\ y\ (u.\,D_1))\ (x.\,E)$
$\equiv\ !\mathsf{L}\ y\ (u.\,\mathsf{cut}\ D_1\ (x.\,E)) = F$ by rule $(\mathsf{cut}/!\mathsf{L}/-)$ [a]

$F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R$ since $D_1 \rightsquigarrow P^x\{u/y\}$ [b]

$\mathsf{cut}\ D_1\ (x.\,E) \equiv F_1$ for some F_1;
$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z)$;
$F_1 \Rightarrow F_1'$;
$F_1' \rightsquigarrow R_1'^z \equiv (\nu x)(P'^x \mid Q'^z)$ by i.h. on D_1, E

$F =\ !\mathsf{L}\ y\ (u.\,F_1) \Rightarrow\ !\mathsf{L}\ y\ (u.\,F_1') = F'$ by congruence [c]
$F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$D = \mathsf{cut}\ D_1\ (y.\,D_2)$ and E arbitrary.

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (\mathsf{cut}\ D_1\ (y.\,D_2))\ (x.\,E)$
$\equiv \mathsf{cut}\ D_1\ (y.\,\mathsf{cut}\ D_2\ (x.\,E)) = F$ by rule $(\mathsf{cut}/-/\mathsf{cut}_1)$ [a]

$F \rightsquigarrow (\nu y)(P_1^y \mid ((\nu x)(P_2^x \mid Q^z))) \equiv (\nu x)((\nu y)(P_1^y \mid P_2^x) \mid Q^z) = R$ [b]

$\mathsf{cut}\ D_2\ (x.\,E) \equiv F_2$ with
$F_2 \rightsquigarrow R_2$ and $R_2 \equiv (\nu x)(P_2^x \mid Q^z)$ and
$F_2 \Rightarrow F_2'$ with
$F_2' \rightsquigarrow R_2' \equiv (\nu x)(P_2'^x \mid Q'^z)$ by i.h. on D_2, E

$F \Rightarrow \mathsf{cut}\ D_1\ (y.\,F_2') = F'$ by congruence [c]
$F' \rightsquigarrow (\nu y)(P_1^y \mid (\nu x)(P_2'^x \mid Q'^z)) \equiv (\nu x)((\nu y)(P_1^y \mid P_2'^x) \mid Q'^z)$
$= (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$D = \mathsf{cut}^! \; D_1 \; (u.\, D_2)$ and E arbitrary.

$\mathsf{cut}\; D\; (x.\, E) = \mathsf{cut}\; (\mathsf{cut}^!\; D_1\; (u.\, D_2))\; (x.\, E)$
$\equiv \mathsf{cut}^!\; D_1\; (u.\, \mathsf{cut}\; D_2\; (x.\, E)) = F$ by rule $(\mathsf{cut}/\mathsf{cut}^!/-)$ [a]

$F \rightsquigarrow (\nu u)((!u(y).P_1^y) \mid (\nu x)(P_2^x \mid Q^z))$
$\equiv (\nu x)((\nu u)((!u(y).P_1^y) \mid P_2^x) \mid Q^z) = R$ by struct. cong. [b]

$\mathsf{cut}\; D_2\; (x.\, E) \equiv F_2$ with
$F_2 \rightsquigarrow R_2$ and $R_2 \equiv (\nu x)(P_2^x \mid Q^z)$ and
$F_2 \Rightarrow F_2'$ with
$F_2' \rightsquigarrow R_2' \equiv (\nu x)(P_2'^x \mid Q'^z)$ by i.h. on D_2, E

$F \Rightarrow \mathsf{cut}^!\; D_1\; (u.\, F_2') = F'$ by congruence [c]

$F' \rightsquigarrow (\nu u)((!u(y).P_1^y) \mid (\nu x)(P_2'^x \mid Q'^z))$
$\equiv (\nu x)((\nu u)((!u(y).P_1^y) \mid P_2'^x) \mid Q'^z) = (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$E = \mathbf{1}\mathsf{L}\; y\; E_1$ and D arbitrary.

$\mathsf{cut}\; D\; (x.\, E) = \mathsf{cut}\; D\; (x.\, \mathbf{1}\mathsf{L}\; y\; E_1)$
$\equiv \mathbf{1}\mathsf{L}\; y\; (\mathsf{cut}\; D\; (x.\, E_1)) = F$ by rule $(\mathsf{cut}/-/\mathbf{1}\mathsf{L})$ [a]

$F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R$ [b]

$\mathsf{cut}\; D\; (x.\, E_1) \equiv F_1$ for some F_1;
$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z)$;
$F_1 \Rightarrow F_1'$;
$F_1' \rightsquigarrow R_1' \equiv (\nu x)(P'^x \mid Q'^z)$ by i.h. on D, E_1

$F = \mathbf{1}\mathsf{L}\; y\; F_1 \Rightarrow \mathbf{1}\mathsf{L}\; y\; F_1' = F'$ by congruence [c]

$F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$E = !\mathsf{L}\; y\; (u.\, E_1)$ and D arbitrary.

$\mathsf{cut}\; D\; (x.\, E) = \mathsf{cut}\; D\; (x.\, !\mathsf{L}\; y\; (u.\, E_1))$
$\equiv\; !\mathsf{L}\; y\; (u.\, \mathsf{cut}\; D\; (x.\, E_1)) = F$ by rule $(\mathsf{cut}/-/!\mathsf{L})$ [a]

$F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R$ since $E_1 \rightsquigarrow Q^z\{u/y\}$ [b]

$\mathsf{cut}\; D\; (x.\, E_1) \equiv F_1$ for some F_1;
$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z)$;
$F_1 \Rightarrow F_1'$;
$F_1' \rightsquigarrow R_1' \equiv (\nu x)(P'^x \mid Q'^z)$ by i.h. on D, E_1

$F = !\mathsf{L}\; y\; (u.\, F_1) \Rightarrow \mathbf{1}\mathsf{L}\; y\; (u.\, F_1') = F'$ by congruence [c]

$F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z)$ [d]

Case:$E = \mathsf{cut}\; E_1\; (y.\, E_2)$ with $x \in FV(E_1)$ and D arbitrary.

$\mathsf{cut}\; D\; (x.\, E) = \mathsf{cut}\; D\; (x.\, \mathsf{cut}\; E_1\; (y.\, E_2))$
$\equiv \mathsf{cut}\; (\mathsf{cut}\; D\; (x.\, E_1))\; (y.\, E_2) = F$ by reduction $(\mathsf{cut}/-/\mathsf{cut}_1)$ [a]

$F \rightsquigarrow (\nu y)((\nu x)(P^x \mid Q_1^y) \mid Q_2^z) \equiv (\nu x)(P^x \mid (\nu y)(Q_1^y \mid Q_2^z)) = R$ [b]

$\mathsf{cut}\; D\; (x.\, E_1) \equiv F_1$ with
$F_1 \rightsquigarrow R_1 \equiv (\nu x)(P^x \mid Q_1^y)$ and

$F_1 \Rightarrow F_1'$ with
$F_1' \rightsquigarrow R_1' \equiv (\nu x)(P'^x \mid Q_1'^y)$ by i.h. on D, E_1

$F \Rightarrow \mathsf{cut}\ F_1'\ (y.\, E_2) = F'$ by congruence [c]
$F' \rightsquigarrow (\nu y)((\nu x)(P'^x \mid Q_1'^y) \mid Q_2^z) \equiv (\nu x)(P'^x \mid (\nu y)(Q_1'^y \mid Q_2^z))$
$= (\nu x)(P'^x \mid Q'^z)$. [d]

Case: $E = \mathsf{cut}\ E_1\ (y.\, E_2)$ with $x \in FV(E_2)$ and D arbitrary.

$\mathsf{cut}\ D\ (x.\, E) = \mathsf{cut}\ D\ (x.\, \mathsf{cut}\ E_1\ (y.\, E_2))$
$\equiv \mathsf{cut}\ E_1\ (y.\, \mathsf{cut}\ D\ (x.\, E_2)) = F$ by reduction ($\mathsf{cut}/-/\mathsf{cut}_2$) [a]

$F \rightsquigarrow (\nu y)(Q_1^y \mid (\nu x)(P^x \mid Q_2^z)) \equiv (\nu x)(P^x \mid (\nu y)(Q_1^y \mid Q_2^z)) = R$ [b]

$\mathsf{cut}\ D\ (x.\, E_2) \equiv F_2$ with
$F_2 \rightsquigarrow R_2 \equiv (\nu x)(P^x \mid Q_2^z)$ and
$F_2 \Rightarrow F_2'$ with
$F_2' \rightsquigarrow R_2' \equiv (\nu x)(P'^x \mid Q_2'^z)$ by i.h. on D, E_2

$F \Rightarrow \mathsf{cut}\ E_1\ (y.\, F_2') = F'$ by congruence [c]
$F' \rightsquigarrow (\nu y)(Q_1^y \mid (\nu x)(P'^x \mid Q_2'^z)) \equiv (\nu x)(P'^x \mid (\nu y)(Q_1^y \mid Q_2'^z))$
$= (\nu x)(P'^x \mid Q'^z)$ [d]

Case: $E = \mathsf{cut}^!\ E_1\ (u.\, E_2)$ with $x \in FV(E_1)$ and D arbitrary. This case is impossible because E_1 contains no free linear variables.

Case: $E = \mathsf{cut}^!\ E_1\ (u.\, E_2)$ with $x \in FV(E_2)$ and D arbitrary.

$\mathsf{cut}\ D\ (x.\, E) = \mathsf{cut}\ D\ (x.\, \mathsf{cut}^!\ E_1\ (u.\, E_2))$
$\equiv \mathsf{cut}^!\ E_1\ (u.\, \mathsf{cut}\ D\ (x.\, E_2)) = F$ by reduction ($\mathsf{cut}/-/\mathsf{cut}^!$) [a]

$F \rightsquigarrow (\nu u)((!u(y).Q_1^y) \mid (\nu x)(P^x \mid Q_2^z)) \equiv$
$(\nu x)(P^x \mid (\nu u)((!u(y).Q_1^y) \mid Q_2^z)) = R$ [b]

$\mathsf{cut}\ D\ (x.\, E_2) \equiv F_2$ with
$F_2 \rightsquigarrow R_2 \equiv (\nu x)(P^x \mid Q_2^z)$ and
$F_2 \Rightarrow F_2'$ with
$F_2' \rightsquigarrow R_2' \equiv (\nu x)(P'^x \mid Q_2'^z)$ by i.h. on D, E_2

$F \Rightarrow \mathsf{cut}^!\ E_1\ (u.\, F_2') = F'$ by congruence [c]
$F' \rightsquigarrow (\nu u)((!u(y).Q_1^y) \mid (\nu x)(P'^x \mid Q_2'^z))$
$\equiv (\nu x)(P'^x \mid (\nu u)((!u(y).Q_1^y) \mid Q_2'^z)) = (\nu x)(P'^x \mid Q'^z)$ [d]

□

Lemma A.5. Assume

a. $\Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x{:}C_1 \oplus C_2$ with $P \xrightarrow{\overline{x.\mathsf{inr}}} P'$;
b. $\Gamma; \Delta_2, x{:}C_1 \oplus C_2 \vdash E \rightsquigarrow Q :: z{:}C$ with $Q \xrightarrow{x.\mathsf{inr}} Q'$.

Then,

c. $\mathsf{cut}\ D\ (x.\, E) \equiv\Rightarrow\equiv F$ for some F;
d. $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu x)(P' \mid Q')$.

Proof. Symmetric to the previous lemma. □

Lemma A.6. Assume

a. $\Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x{:}C_1 \otimes C_2$ with $P \xrightarrow{\overline{(\nu y)x\langle y\rangle}} P'$;
b. $\Gamma; \Delta_2, x{:}C_1 \otimes C_2 \vdash E \rightsquigarrow Q :: z{:}C$ with $Q \xrightarrow{x(y)} Q'$.

Then,

c. $\mathsf{cut}\ D(x.\,E) \equiv\Rightarrow\equiv F$ for some F;
d. $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu x)(P' \mid Q')$.

Proof.
By simultaneous induction on D and E. The possible cases for D are $\otimes\mathsf{R}$, $\mathbf{1}\mathsf{L}$, $!\mathsf{L}$, cut and $\mathsf{cut}^!$. The possible cases for E are $\otimes\mathsf{L}$, $\mathbf{1}\mathsf{L}$, $!\mathsf{L}$, cut and $\mathsf{cut}^!$.

Case:$D = \otimes\mathsf{R}\ D_1\ D_2$ and $E = \otimes\mathsf{L}\ x\ (y.x.\,E')$.

$D_1 \rightsquigarrow P_1^y$ and $D_2 \rightsquigarrow P_2^x$ where
$P^x = (\nu y)(x\langle y\rangle.(P_1^y \mid P_2^x)) \xrightarrow{\overline{(\nu y)x\langle y\rangle}} (\nu y)(P_1^y \mid P_2^x) = P'^x$ by inversion
$E' \rightsquigarrow Q'^z$ where $Q^z = x(y).Q'^z \xrightarrow{x(y)} Q'^z$ by inversion
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (\otimes\mathsf{R}\ D_1\ D_2)\ (\otimes\mathsf{L}\ x\ (y.x.\,E'))$
$\Rightarrow \mathsf{cut}\ D_1\ (y.\,\mathsf{cut}\ D_2\ (x.\,E')) = F$ by rule $(\mathsf{cut}/\otimes\mathsf{R}/\otimes\mathsf{L})$

$F \rightsquigarrow (\nu y)(P_1^y \mid (\nu x)(P_2^x \mid Q'^z)) \equiv (\nu x)((\nu y)(P_1^y \mid P_2^x) \mid Q'^z) = (\nu x)(P'^x \mid Q'^z)$.

Case:$D = \mathbf{1}\mathsf{L}\ n\ D'$ and E arbitrary.

$D' \rightsquigarrow P$ by inversion
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (\mathbf{1}\mathsf{L}\ n\ D')\ (x.\,E)$
$\equiv \mathbf{1}\mathsf{L}\ n\ (\mathsf{cut}\ D'\ (x.\,E))$ by rule $(\mathsf{cut}/\mathbf{1}\mathsf{L}/-)$
$\Rightarrow \mathbf{1}\mathsf{L}\ n\ F' = F$ for some $F' \rightsquigarrow (\nu x)(P' \mid Q')$ by i.h. on D' and E
$F \rightsquigarrow (\nu x)(P' \mid Q')$

Case:$D = !\mathsf{L}\ n\ (u.\,D')$ and E arbitrary.

$D' \rightsquigarrow P$ by inversion
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (!\mathsf{L}\ n\ (u.\,D'))\ (x.\,E)$
$\equiv\ !\mathsf{L}\ n\ (u.\,\mathsf{cut}\ D'\ (x.\,E))$ by rule $(\mathsf{cut}/!\mathsf{L}/-)$
$\Rightarrow\ !\mathsf{L}\ n\ (u.\,F') = F$ for some $F' \rightsquigarrow (\nu x)(P' \mid Q')$ by i.h. on D' and E
$F \rightsquigarrow (\nu x)(P' \mid Q')$

Case:$D = \mathsf{cut}\ D_1\ (n.\,D_2)$ and E arbitrary.

$D_1 \rightsquigarrow P_1^n$ and $D_2 \rightsquigarrow P_2^x$ and $P_2^x \xrightarrow{\overline{(\nu y)x\langle y\rangle}} P_2'^x$ where
$P = (\nu n)(P_1^n \mid P_2^x) \xrightarrow{\overline{(\nu y)x\langle y\rangle}} (\nu n)(P_1^n \mid P_2'^x) = P'^x$ by inversion
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (\mathsf{cut}\ D_1\ (n.\,D_2))\ (x.\,E)$
$\equiv \mathsf{cut}\ D_1\ (n.\,\mathsf{cut}\ D_2\ (x.\,E))$ by rule $(\mathsf{cut}/\mathsf{cut}/-)$
$\Rightarrow \mathsf{cut}\ D_1\ (n.\,F_2) = F$ where $F_2 \rightsquigarrow (\nu x)(P_2'^x \mid Q'^z)$ by i.h. on D_2 and E
$F \rightsquigarrow (\nu n)(P_1^n \mid (\nu x)(P_2'^x \mid Q'^z))$

$\equiv (\nu x)((\nu n)(P_1^n \mid P_2'^x) \mid Q'^z)$
$= (\nu x)(P'^x \mid Q'^z)$

Case: $D = \mathsf{cut}^!\ D_1\ (u.\,D_2)$ and E arbitrary.

$D_1 \rightsquigarrow (!u(n).P_1^n)$ and $D_2 \rightsquigarrow P_2^x$ and $P_2^x \xrightarrow{\overline{(\nu y)x\langle y\rangle}} P_2'^x$ where
$P = (\nu u)((!u(n).P_1^n) \mid P_2^x) \xrightarrow{\overline{(\nu y)x\langle y\rangle}} (\nu u)((!u(n).P_1^n) \mid P_2'^x) = P'^x$ — by inversion

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (\mathsf{cut}^!\ D_1\ (u.\,D_2))\ (x.\,E)$
$\equiv \mathsf{cut}^!\ D_1\ (u.\,\mathsf{cut}\ D_2\ (x.\,E))$ — by rule $(\mathsf{cut}/\mathsf{cut}^!/-)$
$\Rightarrow \mathsf{cut}\ D_1\ (u.\,F_2) = F$ where $F_2 \rightsquigarrow (\nu x)(P_2'^x \mid Q'^z)$ — by i.h. on D_2 and E

$F \rightsquigarrow (\nu u)((!u(n).P_1^n) \mid (\nu x)(P_2'^x \mid Q'^z))$
$\equiv (\nu x)((\nu u)((!u(n).P_1^n) \mid P_2'^x) \mid Q'^z)$
$= (\nu x)(P'^x \mid Q'^z)$

Case: $E = \mathbf{1}\mathsf{L}\ n\ E'$ and D arbitrary.

$E' \rightsquigarrow Q^z$ — by inversion
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\,\mathbf{1}\mathsf{L}\ n\ E')$
$\equiv \mathbf{1}\mathsf{L}\ n\ (\mathsf{cut}\ D\ (x.\,E'))$ — by rule $(\mathsf{cut}/-/\mathbf{1}\mathsf{L})$
$\Rightarrow \mathbf{1}\mathsf{L}\ n\ F' = F$ for some $F' \rightsquigarrow (\nu x)(P'^x \mid Q'^z)$ — by i.h. on D and E'
$F \rightsquigarrow (\nu x)(P'^x \mid Q'^z)$

Case: $E = !\mathsf{L}\ n\ (u.\,E')$ and D arbitrary.

$E' \rightsquigarrow Q^z$ — by inversion
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\,!\mathsf{L}\ n\ (u.\,E')$
$\equiv !\mathsf{L}\ n\ (u.\,\mathsf{cut}\ D\ (x.\,E'))$ — by rule $(\mathsf{cut}/-/!\mathsf{L})$
$\Rightarrow !\mathsf{L}\ n\ (u.\,F') = F$ for some $F' \rightsquigarrow (\nu x)(P'^x \mid Q'^z)$ — by i.h. on D and E'
$F \rightsquigarrow (\nu x)(P'^x \mid Q'^z)$

Case: $E = \mathsf{cut}\ E_1\ (n.\,E_2)$ and $x \in FV(E_1)$.

$E_1 \rightsquigarrow Q_1^n$ and $E_2 \rightsquigarrow Q_2^z$ for $Q_1^n \xrightarrow{x(y)} Q_1'^n$ where
$Q = (\nu n)(Q_1^n \mid Q_2^z) \xrightarrow{x(y)} (\nu n)(Q_1'^n \mid Q_2^z)$ — by inversion

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\,\mathsf{cut}\ E_1\ (n.\,E_2))$
$= \mathsf{cut}\ (\mathsf{cut}\ D\ (x.\,E_1))\ (n.\,E_2)$ — by rule $(\mathsf{cut}/\mathsf{cut}/-)$ and $x \notin FV(E_2)$
$\Rightarrow \mathsf{cut}\ F_1\ (n.\,E_2) = F$ for some $F_1 \rightsquigarrow (\nu x)(P'^x \mid Q_1'^n)$ — by i.h. on D and E_1

$F \rightsquigarrow (\nu n)((\nu x)(P'^x \mid Q_1'^n) \mid Q_2^z)$
$\equiv (\nu x)(P'^x \mid (\nu n)(Q_1'^n \mid Q_2^z))$
$= (\nu x)(P'^x \mid Q'^z)$

Case: $E = \mathsf{cut}\ E_1\ (n.\,E_2)$ and $x \in FV(E_2)$.

$E_1 \rightsquigarrow Q_1^n$ and $E_2 \rightsquigarrow Q_2^z$ for $Q_2^z \xrightarrow{x(y)} Q_2'^z$ where
$Q = (\nu n)(Q_1^n \mid Q_2^z) \xrightarrow{x(y)} (\nu n)(Q_1^n \mid Q_2'^z)$ — by inversion

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\,\mathsf{cut}\ E_1\ (n.\,E_2))$
$= \mathsf{cut}\ E_1\ (n.\,\mathsf{cut}\ D\ (x.\,E_2))$ — by rule $(\mathsf{cut}/-/\mathsf{cut})$ and $x \notin FV(E_1)$
$\Rightarrow \mathsf{cut}\ E_1\ (n.\,F_2) = F$ for some $F_2 \rightsquigarrow (\nu x)(P'^x \mid Q_2'^z)$ — by i.h. on D and E_2

$F \rightsquigarrow (\nu n)(Q_1^n \mid (\nu x)(P'^x \mid Q_2'^z))$
$\equiv (\nu x)(P'^x \mid (\nu n)(Q_1^n \mid Q_2'^z))$
$= (\nu x)(P'^x \mid Q'^z)$

Case:$E = \mathsf{cut}^! \; E_1 \; (u.\, E_2)$

$x \in FV(E_2)$ by inversion

$E_1 \rightsquigarrow (!u(n).Q_1^n)$ and $E_2 \rightsquigarrow Q_2^z$ for $Q_2^z \overset{x(y)}{\rightarrow} Q_2'^z$ where

$Q = (\nu u)((!u(n).Q_1^n) \mid Q_2^z) \overset{x(y)}{\rightarrow} (\nu u)((!u(n).Q_1^n) \mid Q_2'^z)$ by inversion

$\mathsf{cut}\; D\; (x.\, E) = \mathsf{cut}\; D\; (x.\, \mathsf{cut}^!\; E_1\; (u.\, E_2))$
$= \mathsf{cut}^!\; E_1\; (u.\, \mathsf{cut}\; D\; (x.\, E_2))$ by rule $(\mathsf{cut}/-/\mathsf{cut}^!)$
$\Rightarrow \mathsf{cut}^!\; E_1\; (u.\, F_2) = F$ for some $F_2 \rightsquigarrow (\nu x)(P'^x \mid Q_2'^z)$ by i.h. on D and E_2

$F \rightsquigarrow (\nu u)((!u(n).Q_1^n) \mid (\nu x)(P'^x \mid Q_2'^z))$
$\equiv (\nu x)(P'^x \mid (\nu u)((!u(n).Q_1^n) \mid Q_2'^z))$
$= (\nu x)(P'^x \mid Q'^z)$

□

Lemma A.7. Assume

a. $\Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x{:}C_1 \multimap C_2$ with $P \overset{x(y)}{\rightarrow} P'$;

b. $\Gamma; \Delta_2, x{:}C_1 \multimap C_2 \vdash E \rightsquigarrow Q :: z{:}C$ with $Q \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} Q'$.

Then,

c. $\mathsf{cut}\; D(x.\, E) \equiv\Rightarrow\equiv F$ for some F;

d. $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu x)(\nu y)(P' \mid Q')$.

Proof. By simultaneous induction on D and E. The possible cases for D are $\multimap$R, $\mathbf{1}$L, !L, cut and cut$^!$. The possible cases for E are $\multimap$L, $\mathbf{1}$L, !L, cut and cut$^!$.

Case:$D = {\multimap}\mathsf{R}\; (x.\, D_1)$ and $E = {\multimap}\mathsf{L}\; x\; E_1\; (x.\, E_2)$.

$E_1 \rightsquigarrow Q_1^y$ and $E_2 \rightsquigarrow Q_2^z$ where

$Q^z = (\nu y)(x\langle y\rangle.\,(Q_1^y \mid Q_2^z)) \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} (\nu y)(Q_1^y \mid Q_2^z) = Q'^x$ by inversion

$D_1 \rightsquigarrow P'^x$ where $P^x = x(y).P'^x \overset{x(y)}{\rightarrow} P'^x$ by inversion

$\mathsf{cut}\; D\; (x.\, E) = \mathsf{cut}\; ({\multimap}\mathsf{R}\; (x.\, D_1))\; ({\multimap}\mathsf{L}\; x\; E_1\; (x.\, E_2))$
$\Rightarrow \mathsf{cut}\; E_1\; (y.\, \mathsf{cut}\; D_1\; (x.\, E_2)) = F$ by rule $(\mathsf{cut}/{\multimap}\mathsf{R}/{\multimap}\mathsf{L})$

$F \rightsquigarrow (\nu y)(Q_1^y \mid (\nu x)(P_1^x \mid Q_2^z)) \equiv (\nu x)((\nu y)P_1^x \mid (Q_1^y \mid Q_2^z)) = (\nu x)(P'^x \mid Q'^z)$

Case:$D = \mathbf{1}\mathsf{L}\; n\; D'$ and E arbitrary.

$D' \rightsquigarrow P$ by inversion

$\mathsf{cut}\; D\; (x.\, E) = \mathsf{cut}\; (\mathbf{1}\mathsf{L}\; n\; D')\; (x.\, E)$
$\equiv \mathbf{1}\mathsf{L}\; n\; (\mathsf{cut}\; D'\; (x.\, E))$ by rule $(\mathsf{cut}/\mathbf{1}\mathsf{L}/-)$
$\Rightarrow \mathbf{1}\mathsf{L}\; n\; F' = F$ for some $F' \rightsquigarrow (\nu x)(P' \mid Q')$ by i.h. on D' and E

$F \rightsquigarrow (\nu x)(P' \mid Q')$

Case:$D = !\mathsf{L}\; n\; (u.\, D')$ and E arbitrary.

$D' \rightsquigarrow P$ by inversion

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (!\mathsf{L}\ n\ (u.\,D'))\ (x.\,E)$
$\equiv\ !\mathsf{L}\ n\ (u.\,\mathsf{cut}\ D'\ (x.\,E))$ by rule (cut/!L/—)
$\Rightarrow\ !\mathsf{L}\ n\ (u.\,F') = F$ for some $F' \rightsquigarrow (\nu x)(P' \mid Q')$ by i.h. on D' and E

$F \rightsquigarrow (\nu x)(P' \mid Q')$

Case:$D = \mathsf{cut}\ D_1\ (n.\,D_2)$ and E arbitrary.

$D_1 \rightsquigarrow P_1^n$ and $D_2 \rightsquigarrow P_2^x$ and $P_2^x \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} P_2'^x$ where

$P = (\nu n)(P_1^n \mid P_2^x) \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} (\nu n)(P_1^n \mid P_2'^x) = P'^x$ by inversion

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (\mathsf{cut}\ D_1\ (n.\,D_2))\ (x.\,E)$
$\equiv \mathsf{cut}\ D_1\ (n.\,\mathsf{cut}\ D_2\ (x.\,E))$ by rule (cut/cut/—)
$\Rightarrow \mathsf{cut}\ D_1\ (n.\,F_2) = F$ where $F_2 \rightsquigarrow (\nu x)(P_2'^x \mid Q'^z)$ by i.h. on D_2 and E

$F \rightsquigarrow (\nu n)(P_1^n \mid (\nu x)(P_2'^x \mid Q'^z))$
$\equiv (\nu x)((\nu n)(P_1^n \mid P_2'^x) \mid Q'^z)$
$= (\nu x)(P'^x \mid Q'^z)$

Case:$D = \mathsf{cut}^!\ D_1\ (u.\,D_2)$ and E arbitrary.

$D_1 \rightsquigarrow (!u(n).P_1^n)$ and $D_2 \rightsquigarrow P_2^x$ and $P_2^x \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} P_2'^x$ where

$P = (\nu u)((!u(n).P_1^n) \mid P_2^x) \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} (\nu u)((!u(n).P_1^n) \mid P_2'^x) = P'^x$ by inversion

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (\mathsf{cut}^!\ D_1\ (u.\,D_2))\ (x.\,E)$
$\equiv \mathsf{cut}^!\ D_1\ (u.\,\mathsf{cut}\ D_2\ (x.\,E))$ by rule (cut/cut$^!$/—)
$\Rightarrow \mathsf{cut}\ D_1\ (u.\,F_2) = F$ where $F_2 \rightsquigarrow (\nu x)(P_2'^x \mid Q'^z)$ by i.h. on D_2 and E

$F \rightsquigarrow (\nu u)((!u(n).P_1^n) \mid (\nu x)(P_2'^x \mid Q'^z))$
$\equiv (\nu x)((\nu u)((!u(n).P_1^n) \mid P_2'^x) \mid Q'^z)$
$= (\nu x)(P'^x \mid Q'^z)$

Case:$E = \mathbf{1}\mathsf{L}\ n\ E'$ and D arbitrary.

$E' \rightsquigarrow Q^z$ by inversion
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\,\mathbf{1}\mathsf{L}\ n\ E')$
$\equiv \mathbf{1}\mathsf{L}\ n\ (\mathsf{cut}\ D\ (x.\,E'))$ by rule (cut/—/1L)
$\Rightarrow \mathbf{1}\mathsf{L}\ n\ F' = F$ for some $F' \rightsquigarrow (\nu x)(P'^x \mid Q'^z)$ by i.h. on D and E'
$F \rightsquigarrow (\nu x)(P'^x \mid Q'^z)$

Case:$E = !\mathsf{L}\ n\ (u.\,E')$ and D arbitrary.

$E' \rightsquigarrow Q^z$ by inversion
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\ !\mathsf{L}\ n\ (u.\,E')$
$\equiv\ !\mathsf{L}\ n\ (u.\,\mathsf{cut}\ D\ (x.\,E'))$ by rule (cut/—/!L)
$\Rightarrow\ !\mathsf{L}\ n\ (u.\,F') = F$ for some $F' \rightsquigarrow (\nu x)(P'^x \mid Q'^z)$ by i.h. on D and E'
$F \rightsquigarrow (\nu x)(P'^x \mid Q'^z)$

Case:$E = \mathsf{cut}\ E_1\ (n.\,E_2)$ and $x \in FV(E_1)$.

$E_1 \rightsquigarrow Q_1^n$ and $E_2 \rightsquigarrow Q_2^z$ for $Q_1^n \overset{x(y)}{\rightarrow} Q_1'^n$ where

$Q = (\nu n)(Q_1^n \mid Q_2^z) \overset{x(y)}{\rightarrow} (\nu n)(Q_1'^n \mid Q_2^z)$ by inversion

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\,\mathsf{cut}\ E_1\ (n.\,E_2))$
$= \mathsf{cut}\ (\mathsf{cut}\ D\ (x.\,E_1))\ (n.\,E_2)$ by rule $(\mathsf{cut}/\mathsf{cut}/-)$ and $x \notin FV(E_2)$
$\Rightarrow \mathsf{cut}\ F_1\ (n.\,E_2) = F$ for some $F_1 \rightsquigarrow (\nu x)(P'^x \mid Q_1'^n)$ by i.h. on D and E_1

$F \rightsquigarrow (\nu n)((\nu x)(P'^x \mid Q_1'^n) \mid Q_2^z)$
$\equiv (\nu x)(P'^x \mid (\nu n)(Q_1'^n \mid Q_2^z))$
$= (\nu x)(P'^x \mid Q'^z)$

Case:$E = \mathsf{cut}\ E_1\ (n.\,E_2)$ and $x \in FV(E_2)$.

$E_1 \rightsquigarrow Q_1^n$ and $E_2 \rightsquigarrow Q_2^z$ for $Q_2^z \xrightarrow{x(y)} Q_2'^z$ where
$Q = (\nu n)(Q_1^n \mid Q_2^z) \xrightarrow{x(y)} (\nu n)(Q_1^n \mid Q_2'^z)$ by inversion

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\,\mathsf{cut}\ E_1\ (n.\,E_2))$
$= \mathsf{cut}\ E_1\ (n.\,\mathsf{cut}\ D\ (x.\,E_2))$ by rule $(\mathsf{cut}/-/\mathsf{cut})$ and $x \notin FV(E_1)$
$\Rightarrow \mathsf{cut}\ E_1\ (n.\,F_2) = F$ for some $F_2 \rightsquigarrow (\nu x)(P'^x \mid Q_2'^z)$ by i.h. on D and E_2

$F \rightsquigarrow (\nu n)(Q_1^n \mid (\nu x)(P'^x \mid Q_2'^z))$
$\equiv (\nu x)(P'^x \mid (\nu n)(Q_1^n \mid Q_2'^z))$
$= (\nu x)(P'^x \mid Q'^z)$

Case:$E = \mathsf{cut}^!\ E_1\ (u.\,E_2)$

$x \in FV(E_2)$ by inversion
$E_1 \rightsquigarrow (!u(n).Q_1^n)$ and $E_2 \rightsquigarrow Q_2^z$ for $Q_2^z \xrightarrow{x(y)} Q_2'^z$ where
$Q = (\nu u)((!u(n).Q_1^n) \mid Q_2^z) \xrightarrow{x(y)} (\nu u)((!u(n).Q_1^n) \mid Q_2'^z)$ by inversion

$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\,\mathsf{cut}^!\ E_1\ (u.\,E_2))$
$= \mathsf{cut}^!\ E_1\ (u.\,\mathsf{cut}\ D\ (x.\,E_2))$ by rule $(\mathsf{cut}/-/\mathsf{cut}^!)$
$\Rightarrow \mathsf{cut}^!\ E_1\ (u.\,F_2) = F$ for some $F_2 \rightsquigarrow (\nu x)(P'^x \mid Q_2'^z)$ by i.h. on D and E_2

$F \rightsquigarrow (\nu u)((!u(n).Q_1^n) \mid (\nu x)(P'^x \mid Q_2'^z))$
$\equiv (\nu x)(P'^x \mid (\nu u)((!u(n).Q_1^n) \mid Q_2'^z))$
$= (\nu x)(P'^x \mid Q'^z)$

□

Lemma A.8. Assume

a. $\Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x{:}!A$ with $P \xrightarrow{x(y)} P'$;
b. $\Gamma; \Delta_2, x{:}!A \vdash E \rightsquigarrow Q :: z{:}C$ with $Q \xrightarrow{\overline{(\nu y)x\langle y\rangle}} Q'$.

Then,

c. $\mathsf{cut}\ D(x.\,E) \equiv\Rightarrow\equiv F$ for some F;
d. $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu x)(\nu y)(P' \mid Q')$.

Proof. By simultaneous induction on the structure of D, E. There are only five possible cases for D: $!\mathsf{R}\ D_1'$, $\mathbf{1}\mathsf{L}\ n\ D_1'$, $!\mathsf{L}\ n\ (u.\,D_1)$, $\mathsf{cut}\ D_1'\ (n.\,D_1'')$, and $\mathsf{cut}^!\ D_1'\ (u.\,D_1'')$. In all other cases P cannot offer $x(y)$, which follows by analysis of the typed extraction rules and the definition of $\xrightarrow{\alpha}$. There are only four possible cases for E: $!\mathsf{L}\ n\ (u.\,E_2')$, $\mathbf{1}\mathsf{L}\ n\ E_2'$, $\mathsf{cut}\ E_2'\ (n.\,E_2'')$, and $\mathsf{cut}^!\ E_2'\ (u.\,E_2'')$. In all other cases Q cannot offer $\overline{(\nu y)x\langle y\rangle}$, which follows by analysis of the typed extraction rules and the definition of $\xrightarrow{\alpha}$.

Case:$D = {!}\mathsf{R}\ D_1'$, $E = {!}\mathsf{L}\ x\ (u.\ E_2')$.

$\Delta_1 = (\cdot)$	
$\Gamma;\cdot \vdash D_1' \rightsquigarrow R_1 :: u{:}A$ where $P = {!}x(u).R_1$	
$\Gamma, u : A;\Delta_2 \vdash E' \rightsquigarrow Q_1 :: z{:}C$ where $Q = (\nu y)x\langle y\rangle.Q_1'$	by inversion
$\mathsf{cut}\ D\ (x.\,E) =$	this case
$\mathsf{cut}\ ({!}\mathsf{R}\ D_1')\ (x.\ {!}\mathsf{L}\ x\ (u.\ E_2'))$	
$\equiv \mathsf{cut}^!\ D_1'\ (u.\,E_2')$	by (cut/!R/!L)
$\mathsf{cut}^!\ D_1'\ (u.\,E_2') \equiv\Rightarrow\equiv \mathsf{cut}^!\ D_1'\ (u.\,E^*)$ for some E^*	
$\Gamma, u : A;\Delta_2 \vdash E^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(R_1\{y/u\} \mid Q_1')$	by Lemma !Red
Pick $F = \mathsf{cut}^!\ D_1'\ (u.\,E^*)$	[satisfying (c)]
$\Gamma;\Delta \vdash F \rightsquigarrow R :: z{:}C$	by cut
with $R = (\nu x)({!}x(u).R_1 \mid (\nu y)(R_1\{y/u\} \mid Q_1'))$	
$\equiv (\nu y)(\nu x)({!}x(u).P_1 \mid R_1\{y/u\} \mid Q_1')$	
$\equiv (\nu y)(\nu x)(P_1' \mid Q_1')$ since $P_1' \equiv {!}x(u).R_1 \mid R_1\{y/u\}$	[satisfying (d)]

Case:D arbitrary, $E = {!}\mathsf{L}\ y\ (u.\ E_2')$.

$\Delta_2 = (\Delta^*, y : {!}B)$	
$\Gamma, u : B;\Delta^*, x : {!}A \vdash E_2' \rightsquigarrow Q_1 :: z{:}C$	by inversion
$\mathsf{cut}\ D\ (x.\,E_2') \equiv\Rightarrow\equiv E^*$ for some E^*	
$\Gamma, u : B;\Delta^* \vdash E^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(\nu x)(P_1' \mid Q_1')$	by i.h.
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\ {!}\mathsf{L}\ y\ (u.\ E_2'))$	
$\equiv {!}\mathsf{L}\ y\ (u.\ \mathsf{cut}\ D\ (x.\,E_2'))$	by (cut/ $-$ /!L)
$\equiv\Rightarrow\equiv {!}\mathsf{L}\ y\ (u.\ E^*)$	by congruence
Pick $F = {!}\mathsf{L}\ y\ (u.\ E^*)$	[satisfying (c)]
$\Gamma;\Delta \vdash F \rightsquigarrow R :: z{:}C$	by !L
with $R = R'$	[satisfying (d)]

Case:$D = {!}\mathsf{L}\ y\ (u.\ D_1')$, E arbitrary.

$\Delta_1 = (\Delta^*, y : {!}B)$	
$\Gamma, u : B;\Delta^* \vdash D_1' \rightsquigarrow P_1 :: x{:}{!}A$	by inversion
$\mathsf{cut}\ D_1'\ (x.\,E) \equiv\Rightarrow\equiv D^*$ for some D^*	
$\Gamma, u : B;\Delta^* \vdash D^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(\nu x)(P_1' \mid Q_1')$	by i.h.
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ ({!}\mathsf{L}\ y\ (u.\ D_1'))\ (x.\,E)$	
$\equiv {!}\mathsf{L}\ y\ (u.\ \mathsf{cut}\ D_1'\ (x.\,E))$	by (cut/!L/$-$)
$\equiv\Rightarrow\equiv {!}\mathsf{L}\ y\ (u.\ D^*)$	by congruence
Pick $F = {!}\mathsf{L}\ y\ (u.\ D^*)$	[satisfying (c)]
$\Gamma;\Delta \vdash F \rightsquigarrow R :: z{:}C$	by !L
with $R = R'$	[satisfying (d)]

Case:D arbitrary, $E = \mathbf{1}\mathsf{L}\ n\ E_2'$.

$\Delta_2 = (\Delta^*, n : \mathbf{1})$	
$\Gamma, x : A;\Delta^* \vdash E_2' \rightsquigarrow Q_1 :: z{:}C$	by inversion
$\mathsf{cut}\ D\ (x.\,E_2') \equiv\Rightarrow\equiv E^*$ for some E^*	
$\Gamma;\Delta^* \vdash E^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(\nu x)(P_1' \mid Q_1')$	by i.h.

$\mathsf{cut}\ D\ (x.E) = \mathsf{cut}\ D\ (x.\mathbf{1}\mathsf{L}\ n\ E_2')$
$\equiv \mathbf{1}\mathsf{L}\ n\ (\mathsf{cut}\ D\ (x.E_2'))$ by $(\mathsf{cut}/-/\mathbf{1}\mathsf{L})$
$\equiv\Rightarrow\equiv \mathbf{1}\mathsf{L}\ n\ E^*$ by congruence
Pick $F = \mathbf{1}\mathsf{L}\ n\ E^*$ [satisfying (c)]
$\Gamma;\Delta \vdash F \rightsquigarrow R :: z{:}C$ by $\mathbf{1}\mathsf{L}$
with $R = R'$ [satisfying (d)]

Case:$D = \mathbf{1}\mathsf{L}\ n\ D_1'$, E arbitrary.

$\Delta_1 = (\Delta^*, n : \mathbf{1})$
$\Gamma;\Delta^* \vdash D_1' \rightsquigarrow P_1 :: x{:}!A$ by inversion
$\mathsf{cut}\ D_1'\ (x.E) \equiv\Rightarrow\equiv D^*$ for some D^*
$\Gamma;\Delta^* \vdash D^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(\nu x)(P_1' \mid Q_1')$ by i.h.
$\mathsf{cut}\ D\ (x.E) = \mathsf{cut}\ (\mathbf{1}\mathsf{L}\ n.\ D_1')\ (x.E)$
$\equiv \mathbf{1}\mathsf{L}\ n.\ (\mathsf{cut}\ D_1'\ (x.E))$ by $(\mathsf{cut}/\mathbf{1}\mathsf{L}/-)$
$\equiv\Rightarrow\equiv \mathbf{1}\mathsf{L}\ n\ D^*$ by congruence
Pick $F = \mathbf{1}\mathsf{L}\ n\ D^*$ [satisfying (c)]
$\Gamma;\Delta \vdash F \rightsquigarrow R :: z{:}C$ by $\mathbf{1}\mathsf{L}$
with $R = R'$ [satisfying (d)]

Case:D arbitrary, $E = \mathsf{cut}\ E_2'\ (n.\ E_2'')$.

$\Delta_2 = (\Delta_2', \Delta_2'')$
$Q_1 = (\nu n)(R_1 \mid R_2)$
$\Gamma, x : A; \Delta_2' \vdash E_2' \rightsquigarrow R_1 :: n{:}B$
$\Gamma, x : A; \Delta_2'', n : B \vdash E_2'' \rightsquigarrow R_2 :: z{:}C$ by inversion

Subcase: $R_1 \xrightarrow{(\nu y)x\langle y\rangle} R_1'$ and $Q_1' = (\nu n)(R_1' \mid R_2)$
$\mathsf{cut}\ D\ (x.E_2') \equiv\Rightarrow\equiv D^*$ for some D^*
$\Gamma;\Delta_2' \vdash D^* \rightsquigarrow R' :: n{:}B$ with $R' \equiv (\nu y)(\nu x)(P_1' \mid R_1')$ by i.h.
$\mathsf{cut}\ D\ (x.E) = \mathsf{cut}\ D\ (x.\mathsf{cut}\ E_2'\ (n.\ E_2''))$
$\equiv \mathsf{cut}\ (\mathsf{cut}\ D\ (x.\ E_2'))\ (n.E_2'')$ by $(\mathsf{cut}/-/\mathsf{cut}_1)$
$\equiv\Rightarrow\equiv \mathsf{cut}\ D^*\ (n.E_2'')$ by congruence
Pick $F = \mathsf{cut}\ D^*\ (n.E_2'')$ [satisfying (c)]
$\Gamma;\Delta \vdash F \rightsquigarrow R :: z{:}C$ by cut
with $R = (\nu n)(R' \mid R_2)$
$\equiv (\nu n)((\nu y)(\nu x)(P_1' \mid R_1') \mid R_2)$
$\equiv (\nu n)(\nu y)(\nu x)(P_1' \mid R_1' \mid R_2)$
$\equiv (\nu y)(\nu x)(P_1' \mid (\nu n)(R_1' \mid R_2))$
$\equiv (\nu y)(\nu x)(P_1' \mid Q_1')$ [satisfying (d)]

Subcase: $R_2 \xrightarrow{(\nu y)x\langle y\rangle} R_2'$ and $Q_1' = (\nu n)(R_1 \mid R_2')$
$\mathsf{cut}\ D\ (x.E_2'') \equiv\Rightarrow\equiv D^*$ for some D^*
$\Gamma;\Delta_2'', n : B \vdash D^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(\nu x)(P_1' \mid R_2')$ by i.h.
$\mathsf{cut}\ D_1\ (x.D_2) = \mathsf{cut}\ D\ (x.\mathsf{cut}\ E_2'\ (n.\ E_2''))$
$\equiv \mathsf{cut}\ E_2'\ (n.\ \mathsf{cut}\ D\ (x.\ E_2''))$ by $(\mathsf{cut}/-/\mathsf{cut}_2)$
$\equiv\Rightarrow\equiv \mathsf{cut}\ E_2'\ (n.D^*)$ by congruence
Pick $F = \mathsf{cut}\ E_2'\ (n.D^*)$ [satisfying (c)]

$\Gamma;\Delta \vdash F \rightsquigarrow R :: z{:}C$ by cut
with $R = (\nu n)(R_1 \mid R')$
$\equiv (\nu n)(R_1 \mid (\nu y)(\nu x)(P_1' \mid R_2'))$
$\equiv (\nu y)(\nu x)(\nu n)(R_1 \mid P_1' \mid R_2')$
$\equiv (\nu y)(\nu x)(\nu n)(P_1' \mid R_1 \mid R_2')$
$\equiv (\nu y)(\nu x)(P_1' \mid (\nu n)(R_1 \mid R_2'))$
$\equiv (\nu y)(\nu x)(P_1' \mid Q_1')$ [satisfying (d)]

Case:$D = \mathsf{cut}\ D_1'\ (n.\ D_1'')$, E arbitrary.

$\Delta_1 = (\Delta_1', \Delta_1'')$
$P_1 = (\nu n)(R_1 \mid R_2)$
$\Gamma;\Delta_1' \vdash D_1' \rightsquigarrow R_1 :: n{:}B$
$\Gamma;\Delta_1'', n{:}B \vdash D_1'' \rightsquigarrow R_2 :: x{:}!A$ by inversion
$R_2 \overset{x(y)}{\rightarrow} R_2'$ and $P_1' = (\nu n)(R_1 \mid R_2')$
$\mathsf{cut}\ D_1''\ (x.\,E) \equiv\Rightarrow\equiv D^*$ for some D^*
$\Gamma;\Delta_1'', n{:}B, \Delta_2 \vdash D^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(\nu x)(R_2' \mid Q_1')$ by i.h.
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ (\mathsf{cut}\ D_1'\ (n.\ D_1''))\ (x.\ E)$
$\equiv \mathsf{cut}\ D_1'\ (n.\,\mathsf{cut}\ D_1''\ (x.\,E))$ by $(\mathsf{cut}/-/\mathsf{cut}_1)$
$\equiv\Rightarrow\equiv \mathsf{cut}\ D_1'\ (n.\,D^*)$ by congruence
Pick $F = \mathsf{cut}\ D_1'\ (n.\,D^*)$ [satisfying (c)]
$\Gamma;\Delta \vdash F \rightsquigarrow R :: z{:}C$ by cut
with $R = (\nu n)(R_1 \mid R')$
$\equiv (\nu n)(R_1 \mid (\nu y)(\nu x)(R_2' \mid Q_1')$
$\equiv (\nu y)(\nu x)(\nu n)(R_1 \mid R_2' \mid Q_1')$
$\equiv (\nu y)(\nu x)(\nu n)(R_1 \mid R_2') \mid Q_1))$
$\equiv (\nu y)(\nu x)(P_1' \mid Q_1')$ [satisfying (d)]

Case:D arbitrary, $E = \mathsf{cut}^!\ E_2'\ (u.\ E_2'')$.

$Q_1 = (\nu u)(!u(w).R_1 \mid R_2)$
$\Gamma;\cdot \vdash E_2' \rightsquigarrow R_1 :: w{:}B$
$\Gamma, u : B;\Delta_2, x : !A \vdash E_2'' \rightsquigarrow R_2 :: z{:}C$ by inversion
$R_2 \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} R_2'$ and $Q_1' = (\nu u)(!u(w).R_1 \mid R_2')$
$\mathsf{cut}\ D\ (x.\,E_2'') \equiv\Rightarrow\equiv D^*$ for some D^*
$\Gamma, u : B;\Delta_2 \vdash D^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(\nu x)(P_1' \mid R_2')$ by i.h.
$\mathsf{cut}\ D\ (x.\,E) = \mathsf{cut}\ D\ (x.\mathsf{cut}^!\ E_2'\ (u.\ E_2''))$
$\equiv \mathsf{cut}^!\ E_2' (u.\ \mathsf{cut}\ D\ (x.\,E_2''))$ by $(\mathsf{cut}/-/\mathsf{cut}^!)$
$\equiv\Rightarrow\equiv \mathsf{cut}^!\ E_2'\ (u.\,D^*)$ by congruence
Pick $F = \mathsf{cut}^!\ E_2'\ (u.\,D^*)$ [satisfying (c)]
$\Gamma;\Delta \vdash F \rightsquigarrow R :: z{:}C$ by $\mathsf{cut}^!$
with $R = (\nu u)(!u(w).R_1 \mid R')$
$\equiv (\nu u)(!u(w).R_1 \mid (\nu y)(\nu x)(P_1' \mid R_2'))$
$\equiv (\nu y)(\nu x)(\nu u)(!u(w).R_1 \mid P_1' \mid R_2')$

$\equiv (\nu y)(\nu x)(\nu u)(P_1' \mid !u(w).R_1 \mid R_2')$
$\equiv (\nu y)(\nu x)(P_1' \mid (\nu u)(!u(w).R_1 \mid R_2'))$
$\equiv (\nu y)(\nu x)(P_1' \mid Q_1')$ [satisfying (d)]

Case:$D_1 = \mathsf{cut}^! \; D_1' \; (u.\, D_1'')$, E arbitrary.

$P_1 = (\nu u)(!u(w).R_1 \mid R_2)$
$\Gamma \vdash D_1' \rightsquigarrow R_1 :: w{:}B$
$\Gamma, u : B; \Delta_1 \vdash D_1'' \rightsquigarrow R_2 :: x{:}!A$ by inversion
$R_2 \overset{x(y)}{\to} R_2'$ and $P_1' = (\nu u)(!u(w).R_1 \mid R_2')$
$\mathsf{cut}\; D_1'' \; (x.\, E) \equiv\Rightarrow\equiv D^*$ for some D^*
$\Gamma, u : B; \Delta \vdash D^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(\nu x)(R_2' \mid Q_1')$ by i.h.
$\mathsf{cut}\; D_1 \; (x.\, E) = \mathsf{cut}\; (\mathsf{cut}^! \; D_1' \; (u.\, D_1''))\; (x.\, E)$
$\equiv \mathsf{cut}^! \; D_1' \; (u.\, \mathsf{cut}\; D_1'' \; (x.\, E))$ by $(\mathsf{cut}/\mathsf{cut}^!/-)$
$\equiv\Rightarrow\equiv \mathsf{cut}^! \; D_1' \; (u.\, D^*)$ by congruence
Pick $F = \mathsf{cut}^! \; D_1' \; (u.\, D^*)$ [satisfying (c)]
$\Gamma; \Delta \vdash F \rightsquigarrow R :: z{:}C$ by $\mathsf{cut}^!$
with $R = (\nu u)(!u(w).R_1 \mid (\nu y)(\nu x)(R_2' \mid Q_1'))$
$\equiv (\nu y)(\nu x)(\nu u)(!u(w).R_1 \mid R_2' \mid Q_1'))$
$\equiv (\nu y)(\nu x)((\nu u)(!u(w).R_1 \mid R_2') \mid Q_1'))$
$\equiv (\nu y)(\nu x)(P_1' \mid Q_1')$ [satisfying (d)]

□

Lemma A.9. Assume

a. $\Gamma; \cdot \vdash D \rightsquigarrow P \; :: u{:}A$ and
b. $\Gamma, u{:}A; \Delta \vdash E \rightsquigarrow Q :: z{:}C$ with $Q \overset{\overline{(\nu y)u\langle y\rangle}}{\to} Q'$.

Then,

c. $\mathsf{cut}^! \; D_1 \; (u.\, D_2) \equiv\Rightarrow\equiv \mathsf{cut}^! \; D_1 \; (u.\, F)$ for some F;
d. $\Gamma, u{:}A; \Delta \vdash F \rightsquigarrow R :: z{:}C$ with $R \equiv (\nu y)(P\{y/u\} \mid Q')$.

Proof. By induction on the structure of E. There are only five possible cases for E: $\mathbf{1}\mathsf{L}\; n\; E_2'$, $\mathsf{copy}\; u\; (x.\, E_2')$, $!\mathsf{L}\; n\; (u.\, E_2')$, $\mathsf{cut}\; E_2'\; (n.\, E_2'')$ and $\mathsf{cut}^!\; E_2'\; (v.\, E_2'')$. In all other cases Q_1 cannot offer $\overline{(\nu y)u\langle y\rangle}$, which follows by analysis of the typed extraction rules and the definition of $\overset{\alpha}{\to}$.

Case:$E = \mathsf{copy}\; u\; (x.\, E_2')$.

$Q_1 \equiv (\nu y)u\langle y\rangle.Q_1'$
$\Gamma, u : A; \Delta, y : A \vdash E_2' \rightsquigarrow Q_1' :: z : C$ by inversion
$\mathsf{cut}^! \; D\; (u.\, (\mathsf{copy}\; u\; (y.\, E_2'))$
$\Rightarrow \mathsf{cut}\; D\; (y.\, \mathsf{cut}^! \; D\; (u.\, E_2'))$ by $(\mathsf{cut}^!/-/\mathsf{copy})$
$\equiv \mathsf{cut}^! \; D\; (u.\, \mathsf{cut}\; D\; (y.\, E_2'))$ by $(\mathsf{cut}/-/\mathsf{cut}^!)$
Pick $F = \mathsf{cut}\; D\; (y.\, E_2')$ [satisfying (c)]
$\Gamma, u : A; \Delta \vdash F \rightsquigarrow R :: z{:}C$ with $R \equiv (\nu y)(P_1\{y/u\} \mid Q_1')$ [satisfying (d)]

Case:$E = \mathbf{1}\mathsf{L}\ n\ E_2'$.

$\Delta = (\Delta^*, n : \mathbf{1})$

$\Gamma, u{:}A; \Delta^* \vdash E_2' \rightsquigarrow Q_1 :: z{:}C$ with $Q_1 \overset{\overline{(\nu y)u\langle y\rangle}}{\rightarrow} Q_1'$ by inversion

$\mathsf{cut}^!\ D\ (u.\, E_2') \equiv\Rightarrow\equiv \mathsf{cut}^!\ D\ (u.\, D^*)$ for some D^*
$\Gamma; \Delta^* \vdash D^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(P_1\{y/u\} \mid Q_1')$ by i.h.

$\mathsf{cut}^!\ D\,(u.\, \mathbf{1}\mathsf{L}\ n\ E_2')$
$\equiv \mathbf{1}\mathsf{L}\ n\ (\mathsf{cut}^!\ D\,(u.\, E_2'))$ by $(\mathsf{cut}^!/-/\mathbf{1}\mathsf{L})$
$\equiv\Rightarrow\equiv \mathbf{1}\mathsf{L}\ n\ (\mathsf{cut}^!\ D\ (u.\, D^*))$ by congruence
$\equiv \mathsf{cut}^!\ D\ (u.\, \mathbf{1}\mathsf{L}\ n\ D^*)$ by $(\mathsf{cut}^!/-/\mathbf{1}\mathsf{L})$
Pick $F = \mathbf{1}\mathsf{L}\ n\ D^*$ [satisfying (c)]
Pick $R = R'$
$\Gamma, u : A; \Delta \vdash F \rightsquigarrow R :: z{:}C$ with $R \equiv (\nu y)(P_1\{y/u\} \mid Q_1')$ by $\mathbf{1}\mathsf{L}$ [satisfying (d)]

Case:$E = \,!\mathsf{L}\ x\ (v.\, E_2')$.

$\Delta = (\Delta^*, x : !B)$

$\Gamma, v{:}B, u{:}A; \Delta^* \vdash E_2' \rightsquigarrow Q_1 :: z{:}C$ with $Q_1 \overset{\overline{(\nu y)u\langle y\rangle}}{\rightarrow} Q_1'$ by inversion

$\mathsf{cut}^!\ D\ (u.\, E_2') \equiv\Rightarrow\equiv \mathsf{cut}^!\ D\ (u.\, D^*)$ for some D^*
$\Gamma, v{:}B; \Delta^* \vdash D^* \rightsquigarrow R' :: z{:}C$ with $R' \equiv (\nu y)(P_1\{y/u\} \mid Q_1')$ by i.h.

$\mathsf{cut}^!\ D\,(u.\,(!\mathsf{L}\ x\ (v.E_2')))$
$\equiv\ !\mathsf{L}\ x\ (v.\mathsf{cut}^!\ D\,(u.\, E_2'))$ by $(\mathsf{cut}^!/-/!\mathsf{L})$
$\equiv\Rightarrow\equiv\ !\mathsf{L}\ x\ (v.\, \mathsf{cut}^!\ D\ (u.\, D^*))$ by congruence
$\equiv \mathsf{cut}^!\ D\ (u.\, !\mathsf{L}\ x\ (v.\, D^*))$ by $(\mathsf{cut}^!/-/!\mathsf{L})$
Pick $F = \,!\mathsf{L}\ x\ (v.\, D^*))$ [satisfying (a)]
Pick $R = R'$
$\Gamma, u : A; \Delta \vdash F \rightsquigarrow R :: z{:}C$ with $R \equiv (\nu y)(P_1\{y/u\} \mid Q_1')$ by $!\mathsf{L}$ [satisfying (b)]

Case:$E = \mathsf{cut}\ E_2'\ (n.\, E_2'')$.

$\Delta = (\Delta_1, \Delta_2)$
$\Gamma, x : A; \Delta_1 \vdash E_2' \rightsquigarrow R_1 :: n{:}B$
$\Gamma, x : A; \Delta_2, n{:}B \vdash E_2'' \rightsquigarrow R_2 :: z{:}C$ by inversion
$Q_1 = (\nu n)(R_1 \mid R_2)$
Either $R_1 \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} R_1'$ and $Q_1' \equiv (\nu n)(R_1' \mid R_2)$, or
$R_2 \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} R_2'$ and $Q_1' \equiv (\nu n)(R_1 \mid R_2')$
Subcase: $R_1 \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} R_1'$
$\mathsf{cut}^!\ D\ (u.\, E_2') \equiv\Rightarrow\equiv \mathsf{cut}^!\ D\ (u.\, D^*)$ for some D^*
$\Gamma, u : A; \Delta_1 \vdash D^* \rightsquigarrow S :: n{:}B$
with $S \equiv (\nu y)(P_1\{y/u\} \mid R_1')$ by i.h.

$\mathsf{cut}^! \; D \, (u.\, \mathsf{cut} \; E_2' \; (n.\, E_2''))$
$\equiv \mathsf{cut} \, (\mathsf{cut}^! \, D \; (u.\, E_2')) \; (n.\, \mathsf{cut}^! \; D \; (u.\; E_2''))$ — by $(\mathsf{cut}^!/-/\mathsf{cut})$
$\equiv \mathsf{cut} \, (\mathsf{cut}^! \; D \; (u.\, D^*)) \; (n.\, \mathsf{cut}^! \; D \; (u.\; E_2''))$ — by congruence
$\equiv \mathsf{cut}^! \; D \, (u.\, \mathsf{cut} \; D^* \; (n.\, E_2''))$ — by $(\mathsf{cut}^!/-/\mathsf{cut})$
Pick $D = \mathsf{cut} \; D^* \; (n.\, D_2'')$ — [satisfying (c)]
Pick $R = (\nu n)(S \mid R_2)$
$\Gamma, u : A; \Delta \vdash D \leadsto R :: z{:}C$ — by cut
with $R = (\nu n)((\nu y)(P_1\{y/u\} \mid R_1') \mid R_2)$
$\equiv (\nu y)(P_1\{y/u\} \mid (\nu n)(R_1' \mid R_2))$
$\equiv (\nu y)(P_1\{y/u\} \mid Q_1')$ — [satisfying (d)]

Subcase: $R_2 \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} R_2'$
$\mathsf{cut}^! \; D \; (u.\, E_2'') \equiv\Rightarrow\equiv \mathsf{cut}^! \; D \; (u.\, D^*)$ for some D^*
$\Gamma, u : A; \Delta_2, n : B \vdash D^* \leadsto S :: z{:}C$
with $S \equiv (\nu y)(P_1\{y/u\} \mid R_2')$ — by i.h.

$\mathsf{cut}^! \; D \, (u.\, \mathsf{cut} \; E_2' \; (n.\, E_2''))$
$\equiv \mathsf{cut} \, (\mathsf{cut}^! \, D \; (u.\, E_2')) \; (n.\, \mathsf{cut}^! \; D \; (u.\; E_2''))$ — by $(\mathsf{cut}^!/-/\mathsf{cut})$
$\equiv \mathsf{cut} \, (\mathsf{cut}^! \; D \; (u.\, E_2')) \; (n.\, \mathsf{cut}^! \; D \; (u.\; D^*))$ — by congruence
$\equiv \mathsf{cut}^! \; D \, (u.\, \mathsf{cut} \; E_2' \; (n.\, D^*))$ — by $(\mathsf{cut}^!/-/\mathsf{cut})$
Pick $F = \mathsf{cut} \; E_2' \; (n.\, D^*)$ — [satisfying (c)]
Pick $R = (\nu n)(R_1 \mid S)$
$\Gamma, u : A; \Delta \vdash D \leadsto R :: z{:}C$ — by cut
with $R = (\nu n)(R_1 \mid (\nu y)(P_1\{y/u\} \mid R_2'))$
$\equiv (\nu y)(P_1\{y/u\} \mid (\nu n)(R_1 \mid R_2'))$
$\equiv (\nu y)(P_1\{y/u\} \mid Q_1')$. — [satisfying (d)]

Case:$E = \mathsf{cut}^! \; E_2' \; (v.\, E_2'')$.

$\Gamma, u : A; \vdash E_2' \leadsto R_1 :: w{:}B$
$\Gamma, v{:}B, u : A; \Delta_2, \vdash E_2'' \leadsto R_2 :: z{:}C$ — by inversion
$Q_1 = (\nu v)(!v(w).R_1 \mid R_2)$
$R_2 \overset{\overline{(\nu y)x\langle y\rangle}}{\rightarrow} R_2'$ and $Q_1' \equiv (\nu v)(!v(w).R_1 \mid R_2')$
$\mathsf{cut}^! \; D \; (v.\, E_2'') \equiv\Rightarrow\equiv \mathsf{cut}^! \; D \; (v.\, D^*)$ for some D^*
$\Gamma, v : B, u : A; \Delta_2 \vdash D^* \leadsto S :: z{:}C$
with $S \equiv (\nu y)(P_1\{y/u\} \mid R_2')$ — by i.h.

$\mathsf{cut}^! \; D \, (u.\, \mathsf{cut}^! \; E_2' \; (v.\, E_2''))$
$\equiv \mathsf{cut}^! \; (\mathsf{cut}^! \, D \; (u.\, E_2')) \; (v.\, \mathsf{cut}^! \; D \; (u.\; E_2''))$ — by $(\mathsf{cut}^!/-/\mathsf{cut}^!)$
$\equiv \mathsf{cut}^! \; (\mathsf{cut}^! \, D \; (u.\, E_2')) \; (v.\, \mathsf{cut}^! \; D \; (u.\; D^*))$ — by congruence
$\equiv \mathsf{cut}^! \; D \, (u.\, \mathsf{cut}^! \; E_2' \; (v.\, D^*))$ — by $(\mathsf{cut}^!/-/\mathsf{cut}^!)$

Pick $F = \mathsf{cut}^! \; E_2' \; (v.\, D^*)$ — [satisfying (c)]
$\Gamma, u : A; \Delta \vdash F \leadsto R :: z{:}C$ — by cut

with $R = (\nu v)(!v(w).R_1 \mid S)$
$\equiv (\nu v)(!v(w).R_1 \mid (\nu y)(P_1\{y/u\} \mid R_2'))$
$\equiv (\nu y)(\nu v)(!v(w).R_1 \mid P_1\{y/u\} \mid R_2')$
$\equiv (\nu y)(\nu v)(P_1\{y/u\} \mid !v(w).R_1 \mid R_2')$
$\equiv (\nu y)(P_1\{y/u\} \mid (\nu v)(!v(w).R_1 \mid R_2'))$
$\equiv (\nu y)(P_1\{y/u\} \mid Q_1')$ [satisfying (d)]

□

Lemma A.10. Assume

a. $\Gamma; \cdot \vdash D \rightsquigarrow P :: u{:}A$ and

b. $\Gamma, u{:}A; \Delta_2 \vdash E \rightsquigarrow Q :: z{:}C$ with $Q \overset{\overline{(\nu y)u\langle y\rangle}}{\rightarrow} Q'$.

Then,

c. $\mathsf{cut}^! \; D \; (u.\, E) \equiv\Rightarrow\equiv F$ for some F and

d. $\Gamma; \Delta \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu u)(!u(x).P \mid (\nu y)(P\{y/u\} \mid Q'))$.

Proof. Directly from Lemma A.9 and the typed extraction rule for $\mathsf{cut}^!$. □

Theorem A.1. Assume $\Gamma; \Delta \vdash D \rightsquigarrow P :: z{:}A$ and $P \rightarrow Q$.
Then there is E such that $D \equiv\Rightarrow\equiv E$ and $\Gamma; \Delta \vdash E \rightsquigarrow Q :: z{:}A$

Proof. By induction on the structure of D. The possible cases for D are $\mathbf{1}\mathsf{L}$, $!\mathsf{L}$, cut, and $\mathsf{cut}^!$. In all other cases P cannot offer τ.

Case:$D = \mathbf{1}\mathsf{L}\; n\; D'$.

$\Delta = (\Delta^*, n : \mathbf{1})$
$\Gamma; \Delta^* \vdash D' \rightsquigarrow P :: z{:}A$ by inversion
$\Gamma; \Delta^* \vdash E' \rightsquigarrow Q :: z{:}A$ for some E' with $D' \equiv\Rightarrow\equiv E'$ by i.h.
Pick $E = \mathbf{1}\mathsf{L}\; n\; E'$.
$D \equiv\Rightarrow\equiv E$ by congruence
$\Gamma; \Delta \vdash E \rightsquigarrow Q :: z{:}A$ by $\mathbf{1}\mathsf{L}$

Case:$D = !\mathsf{L}\; x\; (u.\; D')$.

$\Delta = (\Delta^*, x : !B)$
$\Gamma, u : B; \Delta^* \vdash D' \rightsquigarrow P :: z{:}A$ by inversion
$\Gamma, u : B; \Delta^* \vdash E' \rightsquigarrow Q :: z{:}A$ for some E' with $D' \equiv\Rightarrow\equiv E'$ by i.h.
Pick $E = !\mathsf{L}\; x\; (u.\; E')$.
$D \equiv\Rightarrow\equiv E$ by congruence
$\Gamma : \Delta \vdash E \rightsquigarrow Q :: z{:}A$ by $\mathbf{1}\mathsf{L}$

Case:$D = \mathsf{cut}^!\; D_1\; (u.\, D_2)$.

$P \equiv (\nu u)(!u(w).P_1 \mid P_2)$
$\Gamma; \vdash D_1 \rightsquigarrow P_1 :: u{:}C$
$\Gamma, u : C; \Delta \vdash D_2 \rightsquigarrow P_2 :: z{:}A$ by inversion
From $P \rightarrow Q$ either
(1) $P_2 \rightarrow Q_2$ and $Q = (\nu u)(!u(w).P_1 \mid Q_2)$

(2) $P_2 \xrightarrow{\overline{(\nu y)u\langle y\rangle}} Q_2$ and $Q = (\nu u(!u(w).P_1 \mid (\nu y)(P_1\{y/u\} \mid Q_2))$

Subcase (1):

$\Gamma, u : C; \Delta \vdash D \rightsquigarrow Q_2 :: z{:}A$ for some E' with $D_2 \equiv\Rightarrow\equiv E$ by i.h.

$\mathsf{cut}^! \ D_1 \ (u. D_2) \equiv\Rightarrow\equiv \mathsf{cut}^! \ D_1 \ (u. E')$

Pick $E = \mathsf{cut}^! \ D_1 \ (u. E')$

$\Gamma; \Delta \vdash E \rightsquigarrow Q :: z{:}A$ by $\mathsf{cut}^!$

Subcase (2):

$\mathsf{cut}^! \ D_1 \ (u. D_2) \equiv\Rightarrow\equiv E$ for some E

$\Gamma; \Delta \vdash E \rightsquigarrow R :: z{:}A$ with $R \equiv Q$ by Corollary A.10

Case:$D = \mathsf{cut} \ D_1 \ (x. D_2)$.

$P \equiv (\nu x)(P_1 \mid P_2)$

$\Delta = (\Delta_1, \Delta_2)$

$\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow P_1 :: x{:}C$

$\Gamma; \Delta_2, x : C \vdash D_2 \rightsquigarrow P_1 :: z{:}A$ by inversion

Since $P \to Q$ there are four subcases:

(1) $P_1 \to Q_1$ and $Q = (\nu x)(Q_1 \mid P_2)$

(2) $P_2 \to Q_2$ and $Q = (\nu x)(P_1 \mid Q_2)$

(3) $P_1 \xrightarrow{\alpha} Q_1$ and $P_2 \xrightarrow{\overline{\alpha}} Q_2$

(4) $P_1 \xrightarrow{\overline{\alpha}} Q_1$ and $P_2 \xrightarrow{\alpha} Q_2$

Subcase (1): $P_1 \to Q_1$

$D_1 \equiv\Rightarrow\equiv E_1$ for some E_1

$\Gamma; \Delta_1 \vdash E_1 \rightsquigarrow Q_1 :: x{:}C$ by i.h.

$D = \mathsf{cut} \ D_1 \ (x. D_2)$

$\equiv\Rightarrow\equiv \mathsf{cut} \ E_1 \ (x. D_2)$ by congruence

Pick $E = \mathsf{cut} \ E_1 \ (x. D_2)$

$\Gamma; \Delta \vdash E \rightsquigarrow Q :: z{:}A$

Subcase (2): $P_2 \to Q_2$

Symmetric to Subcase (1).

Subcase (3): $P_1 \xrightarrow{\alpha} Q_1$ and $P_2 \xrightarrow{\overline{\alpha}} Q_2$

Subsubcase: $C = \mathbf{1}$

not possible

Subsubcase: $C = C_1 \mathbin{\&} C_2$

$\alpha = x.\mathsf{inl}$ or $\alpha = x.\mathsf{inr}$ by Lemma A.1

$\mathsf{cut} \ D_1 \ (x. D_2) \equiv\Rightarrow\equiv D$ for some D

$\Gamma; \Delta \vdash D \rightsquigarrow R :: z : C$

with $R \equiv (\nu x)(Q_1 \mid Q_2) = Q$ by Lemmas A.2 and A.3

Subsubcase: $C = C_1 \oplus C_2$

not possible

Subsubcase: $C = C_1 \otimes C_2$

not possible

Subsubcase: $C = C_1 \multimap C_2$
$\alpha = x(y)$ and $\overline{\alpha} = \overline{(\nu y)x\langle y\rangle}$ by Lemma A.1
$\mathsf{cut}\ D_1\ (x.\,D_2) \equiv\Rightarrow\equiv D$ for some D
$\Gamma; \Delta \vdash D \rightsquigarrow Q :: z : C$
with $R \equiv (\nu x)(\nu y)(Q_1 \mid Q_2) = Q$ by Lemma A.7

Subsubcase: $C = {!C_1}$
$\alpha = x(y)$ and $\overline{\alpha} = \overline{(\nu y)x\langle y\rangle}$ by Lemma A.1
$\mathsf{cut}\ D_1\ (x.\,D_2) \equiv\Rightarrow\equiv D$ for some D
$\Gamma; \Delta \vdash D \rightsquigarrow Q :: z : C$
with $R \equiv (\nu x)(\nu y)(Q_1 \mid Q_2) = Q$ by Lemma A.8

Subcase (4): $P_1 \xrightarrow{\overline{\alpha}} Q_1$ and $P_2 \xrightarrow{\alpha} Q_2$
Subsubcase: $C = \mathbf{1}$
not possible

Subsubcase: $C = C_1 \mathbin{\&} C_2$
not possible

Subsubcase: $C = C_1 \oplus C_2$
$\overline{\alpha} = \overline{x.\mathsf{inl}}$ or $\overline{\alpha} = \overline{x.\mathsf{inr}}$ by Lemma A.1
$\mathsf{cut}\ D_1\ (x.\,D_2) \equiv\Rightarrow\equiv D$ for some D
$\Gamma; \Delta \vdash D \rightsquigarrow R :: z : C$
with $R \equiv (\nu x)(Q_1 \mid Q_2) = Q$ by Lemmas A.4 and A.5

Subsubcase: $C = C_1 \otimes C_2$
$\overline{\alpha} = \overline{(\nu y)x\langle y\rangle}$ and $\alpha = x(y)$ by Lemma A.1
$\mathsf{cut}\ D_1\ (x.\,D_2) \equiv\Rightarrow\equiv D$ for some D
$\Gamma; \Delta \vdash D \rightsquigarrow Q :: z : C$
with $R \equiv (\nu x)(\nu y)(Q_1 \mid Q_2) = Q$

Subsubcase: $C = C_1 \multimap C_2$
not possible

Subsubcase: $C = {!C_1}$
not possible

□

A.3. *Progress lemmas*

Lemma A.11. Assume $\Gamma, \Delta \vdash D \rightsquigarrow P :: z : C$ and not $\mathit{live}(P)$; then

1. $C = \mathbf{1}$ or $C = {!C'}$ for some C'.
2. $(x : A_i) \in \Delta$ implies $A_i = \mathbf{1}$ or there is B_i with $A_i = {!B_i}$.
3. $C = {!C'}$ implies $P \equiv (\nu\overline{x})({!z(y)}.R \mid R')$.

Proof. By structural induction on the structure of D. The only possible cases for D are $\mathbf{1}\mathsf{R}$, $\mathbf{1}\mathsf{L}\ D'$, $!\mathsf{L}\ x\ (u.\,D')$, $\mathsf{cut}\ D'\ (x.\ D'')$, $\mathsf{R!}\ D'$ and $\mathsf{cut}^{!}\ D'\ (x.\ D'')$, which follows by analysis of the typing rules.

Case:$D = \mathsf{cut}\ D'\ (x.\ D'')$

$\Delta = (\Delta_1, D_2)$
$P \equiv (\nu x)(P_1 \mid P_2)$
$\Gamma; \Delta_1 \vdash D' \rightsquigarrow P_1 :: x{:}A$
$\Gamma; \Delta_2, x : A \vdash D'' \rightsquigarrow P_2 :: z{:}C$ — by inversion
not $live(P_1)$ and not $live(P_2)$ — since not $live(P_2)$

$C = \mathbf{1}$ or $C = !C'$ for some C' — [satisfying (1)]
$(x_i : A_i) \in (\Delta_2, x : A)$ implies $A_i = \mathbf{1}$ or there is B_i with $A_i = !B_i$
$C = !C'$ implies $P_2 \equiv (\nu\overline{n})(!z(y).P_2' \mid P_2''')$ — by i.h.

$A = \mathbf{1}$ or $A = !A'$ for some A'
$(x_i : A_i) \in \Delta_1$ implies $A_i = \mathbf{1}$ or there is B_i with $A_i = !B_i$
$A = !A'$ implies $P_1 \equiv (\nu\overline{m})(!x(y).P_1' \mid P_1''')$ — by i.h.

$(x_i : A_i) \in (\Delta_1, \Delta_2) = \Delta$ implies $A_i = \mathbf{1}$
or there is B_j with $A_i = !B_i$ — [satisfying (2)]

$C = !C'$ implies $P \equiv (\nu x)(P_1 \mid P_2)$
$\equiv (\nu x)((\nu\overline{m})(!x(y).P_1' \mid P_1'') \mid (\nu\overline{n})(!z(y).P_2' \mid P_2''))$
$\equiv (\nu x)(\nu\overline{m})(\nu\overline{n})(!x(y).P_1' \mid P_1'' \mid !z(y).P_2' \mid P_2'')$
$\equiv (\nu x)(\nu\overline{m})(\nu\overline{n})(z(y).R \mid R')$
with $R = P_2'$ and $R' = !x(y).P_1' \mid P_1'' \mid !z(y).P_2' \mid P_2''$ — [satisfying (3)]

Case:$D = \mathsf{cut}^!\ D'\ (u.\ D'')$

$P \equiv (\nu x)(!x(y).P_1 \mid P_2)$
$\Gamma; \vdash D' \rightsquigarrow P_1 :: y{:}A$
$\Gamma, u : A; \Delta \vdash D'' \rightsquigarrow P_2 :: z{:}C$ — by inversion
not $live(P_2)$ — since not $live(P_2)$

$C = \mathbf{1}$ or $C = !C'$ for some C' — [satisfying (1)]
$(x_i : A_i) \in \Delta$ implies $A_i = \mathbf{1}$ or there is B_i with $A_i = !B_i$ — [satisfying (2)]
$C = !C'$ implies $P_2 \equiv (\nu\overline{n})(!z(y).P_2' \mid P_2'')$ — by i.h.

$C = !C'$ implies $P \equiv (\nu x)(!x(y).P_1 \mid P_2)$
$\equiv (\nu x)(!x(y).P_1 \mid (\nu\overline{n})(!z(y).P_2' \mid P_2''))$
$\equiv (\nu x)(\nu\overline{n})(!x(y).P_1 \mid !z(y).P_2' \mid P_2''))$
$\equiv (\nu x)(\nu\overline{n})(z(y).R \mid R')$
with $R = P_2'$ and $R' = !x(y).P_1 \mid P_2''$ — [satisfying (3)]

$\square$

Lemma A.12. Let $\Gamma; \Delta \vdash D \rightsquigarrow P :: z : C$. If $live(P)$ then there is Q such that one of the following holds:

a. $P \to Q$,
b. $P \xrightarrow{\alpha} Q$ for some α where $s(\alpha) \in z, \Gamma, \Delta$ and $s(\alpha) \in \Gamma, \Delta$ if $C = !A$.

Proof. By induction on the structure of D. All cases are possible for D except $\mathbf{1}\mathsf{R}$ and $!\mathsf{R}\ D'$, which follows by analysis of the typing rules.

Case:$D = \mathbf{1}\mathsf{L}\ n\ D'$

$\Delta = (\Delta^*, n : \mathbf{1})$
$\Gamma; \Delta^* \vdash D' \rightsquigarrow P :: z{:}C$ by inversion
There is Q such that either $P \to Q$, or $P \xrightarrow{\alpha} Q$
for some α with $s(\alpha) \in z, \Gamma, \Delta^*$ and $s(\alpha) \in \Gamma, \Delta^*$ if $C = !A$. by i.h.
There is Q such that either $P \to Q$, or $P \xrightarrow{\alpha} Q$
for some α with $s(\alpha) \in z, \Gamma, \Delta$ and $s(\alpha) \in \Gamma, \Delta$ if $C = !A$.

Case:$D = !\mathsf{L}\ n\ (u.\ D')$

$\Delta = (n : !A, \Delta^*)$
$\Gamma, u : A; \Delta^* \vdash D' \rightsquigarrow P :: z{:}C$ by inversion
There is Q such that either $P \to Q$, or $P \xrightarrow{\alpha} Q$
for some α with $s(\alpha) \in z, \Gamma, u, \Delta^*$ and $s(\alpha) \in \Gamma, u, \Delta^*$ if $C = !A$. by i.h.
There is Q such that either $P \to Q$, or $P \xrightarrow{\alpha} Q$
for some α with $s(\alpha) \in z, \Gamma, \Delta$ and $s(\alpha) \in \Gamma, \Delta$ if $C = !A$.

Case:$D = \otimes\mathsf{R}\ D_1\ D_2$

$\Delta = (\Delta_1, \Delta_2)$, $C = C_1 \otimes C_2$.
$\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow Q :: y{:}C_1$
$\Gamma; \Delta_2 \vdash D_2' \rightsquigarrow R :: z{:}C_2$
$P \equiv (\nu y) z\langle y\rangle.(Q \mid R)$ by inversion
$P \xrightarrow{(\nu y)\overline{z\langle y\rangle}} Q$ with $z \in z, \Gamma, \Delta$ and $C \neq !A$.

Case:$D = \otimes\mathsf{L}\ (y.\ D_1)$

$\Delta = (D{*}, x : C_1 \otimes C_2)$
$\Gamma; \Delta^*, y : C_1, x : C_2 \vdash D_1 \rightsquigarrow Q :: z{:}C$
$P \equiv x(y).Q$ by inversion
$P \xrightarrow{x(y)} Q$ with $x \in \Gamma, \Delta$

Case:$D = \multimap\mathsf{R}\ D_1$

$C = C_1 \multimap C_2$.
$\Gamma; \Delta, y : C_1 \vdash D_1 \rightsquigarrow Q :: z{:}C_2$
$P \equiv z(y).Q$ by inversion
$P \xrightarrow{z(y)} Q$ with $z \in z, \Gamma, \Delta$ and $C \neq !A$.

Case:$D = \multimap\mathsf{L}\ D_1\ D_2$

$\Delta = (\Delta_1, \Delta_2, x : C_1 \multimap C_2)$
$\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow Q :: y{:}C_1$
$\Gamma; \Delta_2, x : C_2 \vdash D_2' \rightsquigarrow R :: z{:}C_2$
$P \equiv (\nu y) x\langle y\rangle.(Q \mid R)$ by inversion
$P \xrightarrow{(\nu y)\overline{x\langle y\rangle}} Q$ with $x \in \Gamma, \Delta$

Case:$D = \mathsf{cut}\ D_1\ (x.\ D_2)$

$\Delta = (\Delta_1, \Delta_2)$
$\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow P_1 :: x{:}A$
$\Gamma; \Delta_2, x : A \vdash D_2 \rightsquigarrow P_2 :: z{:}C$
$P \equiv (\nu x)(P_1 \mid P_2)$ by inversion
$live(P_1)$ or $live(P_2)$ since $live(P)$

Case (1): $live(P_1)$ and $live(P_2)$.
There is P_1' such that either $P_1 \to P_1'$, or $P_1 \xrightarrow{\alpha_1} P_1'$
for some α_1 with $s(\alpha_1) \in x, \Gamma, \Delta_1$ and $s(\alpha_1) \in \Gamma, \Delta_1$ if $A = !B_1$.
There is P_2' such that either $P_2 \to P_2'$, or $P_2 \xrightarrow{\alpha_2} P_2'$
for some α_2 with $s(\alpha_2) \in x, \Gamma, \Delta_2, z$ and $s(\alpha_2) \in x, \Gamma, \Delta_2$ if $C = !B_2$. by i.h.

Subcase (0.1): $P_1 \to P_1'$ or $P_2 \to P_2'$
$P \to Q$ [satisfying (a)]

Subcase (1.1): $s(\alpha_1) \neq x$
$P \xrightarrow{\alpha_1} Q \equiv (\nu x)(P_1' \mid P_2)$ with $\alpha_1 \in \Gamma, \Delta$ [satisfying (b)]

Subcase (1.2): $s(\alpha_2) \neq x$
$P \xrightarrow{\alpha_2} Q \equiv (\nu x)(P_1 \mid P_2')$ with $\alpha_2 \in z, \Gamma, \Delta$ [satisfying (b)]

Subcase (1.3): $s(\alpha_1) = s(\alpha_2) = x$
$\alpha_2 = \overline{\alpha_1}$ by Lemma A.1
$P \to Q$ with $Q \equiv (\nu x)(\nu y)(P_1' \mid P_2')$ or $Q \equiv (\nu x)(P_1' \mid P_2')$ [satisfying (a)]

Case (2): not $live(P_1)$ and $live(P_2)$
There is P_2' such that either $P_2 \to P_2'$, or $P_2 \xrightarrow{\alpha_2} P_2'$
for some α_2 with $s(\alpha_2) \in x, \Gamma, \Delta_2, z$ and $s(\alpha_2) \in x, \Gamma, \Delta_2$ if $C = !B_2$ by i.h.

Subcase (2.1): $P_2 \to P_2'$
$P \to Q$ with $Q \equiv (\nu x)(P_1 \mid P_2')$ [satisfying (a)]

Subcase (2.2): $P_2 \xrightarrow{\alpha_2} P_2'$
Subcase (2.2.1): $s(\alpha_2) \neq x$
$P \xrightarrow{\alpha_2} Q$ with $Q \equiv (\nu x)(P_1 \mid P_2')$ [satisfying (b)]

Subcase (2.2.2): $s(\alpha_2) = x$
$A \neq \mathbf{1}$ by Lemma A.1
$P_1 \equiv (\nu \overline{y})(!x(w).R_1' \mid R_1'')$
$A = !B$ by Lemma A.11
$P_1 \xrightarrow{x(y)} P_1$
$\alpha_2 = (\nu y)\overline{x\langle w\rangle}$ by Lemma A.1
$P \to Q$ with $Q \equiv (\nu x)(\nu y)(P_1 \mid P_2')$ [satisfying (a)]

Case (3): $live(P_1)$ and not $live(P_2)$
There is P_1' such that either $P_1 \to P_1'$, or $P_1 \xrightarrow{\alpha_1} P_1'$
for some α_1 with $s(\alpha_1) \in \Gamma, \Delta_1, x$ and $s(\alpha_1) \in \Gamma, \Delta_1$ if $C = !B_2$ by i.h.
Subcase (3.1): $P_1 \to P_1'$
$P \to Q$ with $Q \equiv (\nu x)(P_1' \mid P_2)$ [satisfying (a)]

Subcase (3.1): $P_1 \xrightarrow{\alpha_1} P_1'$
for some α_1 with $s(\alpha_1) \in \Gamma, \Delta_1, x$ and $s(\alpha_1) \in \Gamma, \Delta_1$ if $A = !B$ for some B
Subcase (3.1.1) $s(\alpha_1) = x$
$A = \mathbf{1}$ or $A = !B$ for some B by Lemma A.11
Subcase (3.1.1.1) $A = \mathbf{1}$
Impossible, since $P_1 \xrightarrow{\alpha_1} P_1'$ and $s(\alpha_1) = x$ by Lemma A.1
Subcase (3.1.1.2) $A = !B$ for some B
Impossible, since $s(\alpha_1) = x$ contradicts $x \in \Gamma, \Delta_1$.

Case:$D = \mathsf{cut}^! \; D_1 \; (u.\; D_2)$

$\Gamma; \vdash D_1 \rightsquigarrow P_1 :: y{:}A$
$\Gamma, u : A; \Delta \vdash D_2 \rightsquigarrow P_2 :: z{:}C$
$P \equiv (\nu u)(!u(y).P_1 \mid P_2)$ by inversion
$live(P_2)$ since $live(P)$
There is P_2' such that either $P_2 \rightarrow P_2'$, or $P_2 \xrightarrow{\alpha_2} P_2'$
for some α_2 with $s(\alpha_2) \in u, \Gamma, \Delta, z$ and $s(\alpha_2) \in u, \Gamma, \Delta$ if $C = !B$.

Subcase (1): $P_2 \rightarrow P_2'$
$P \rightarrow Q$ with $Q \equiv (\nu u)(P_1 \mid P_2')$ [satisfying (a)]

Subcase (2): $P_2 \xrightarrow{\alpha_2} P_2'$
Subcase (2.1): $s(\alpha_2) \neq u$
$P \xrightarrow{\alpha_2} Q$ with $Q \equiv (\nu u)(P_1 \mid P_2')$
where $s(\alpha_2) \in \Gamma, \Delta, z$ and $s(\alpha_1) \in \Gamma, \Delta$ if $C = !B$ [satisfying (b)]
Subcase (2.2): $s(\alpha_2) = u$
$P_2 \xrightarrow{\overline{(\nu y)u\langle y \rangle}} P_2'$ by Lemma A.1
$!u(y).P_1 \xrightarrow{u(y)} (P_1 \mid !u(y).P_1)$
$P \rightarrow Q$ with $Q \equiv (\nu x)(\nu y)(P_1 \mid !u(y).P_1 \mid P_2')$ [satisfying (a)]

□

Theorem A.2. Let $\cdot;\cdot \vdash D \rightsquigarrow P :: x : \mathbf{1}$, then either P is terminated, P is a composition of replicated servers or there exists Q st. $P \rightarrow Q$.

Proof. If the typing derivation consists of $\mathbf{1}\mathsf{R}$ then we are done. If the typing derivation consists solely of instances of $\mathsf{cut}^!$ then we are done since P is a composition of replicated servers. Otherwise we note that our definition of *live* process accounts for the remaining cases and thus the following reasoning applies: (a) $P \rightarrow Q$, or (b) $P \xrightarrow{\alpha} Q$ for some α where $s(\alpha) = x$, by Lemma A.12. But $P \xrightarrow{\alpha} Q$ with $s(\alpha) = x$ is not possible since $x : \mathbf{1}$, by Lemma A.1. So $P \rightarrow Q$. □

A.4. *Intuitionistic duality*

Proposition A.3. Let A be a !-free type. Then $\Gamma; \Delta \vdash P :: x{:}A$ implies $\Gamma; \Delta, x{:}\overline{A} \vdash P :: -{:}\mathbf{1}$.

Proof. We make the proof term explicit, writing $\Gamma ; \Delta \vdash D \rightsquigarrow P :: x{:}A$, and proceed by induction on the structure of D, constructing $\Gamma ; \Delta, x{:}\overline{A} \vdash P :: -{:}\mathbf{1}$ in each case (eliding a D'). We show a few representative cases.

Case:$D = \mathbf{1}\mathsf{R}$

$\Delta = \cdot$ and $P = \mathbf{0}$ by inversion
$\Gamma ; x{:}\mathbf{1} \vdash \mathbf{0} :: -{:}\mathbf{1}$ by $\mathbf{1}\mathsf{R}$ and $\mathbf{1}\mathsf{L}$

Case:$D = \otimes\mathsf{R}\ D_1\ D_2$

$\Delta = (\Delta_1, \Delta_2)$, $P = (\nu y)x\langle y\rangle.(P_1 \mid P_2)$, $A = A_1 \otimes A_2$,
$\Gamma ; \Delta_1 \vdash D_1 \rightsquigarrow P_1 :: y{:}A_1$
$\Gamma ; \Delta_2 \vdash D_2 \rightsquigarrow P_2 :: x{:}A_2$ by inversion
$\Gamma ; \Delta_2, x{:}\overline{A_2} \vdash P_2 :: -{:}\mathbf{1}$ by i.h. on D_2
$\Gamma ; \Delta_1, \Delta_2, x{:}A_1 \multimap \overline{A_2} \vdash (\nu y)x\langle y\rangle.(P_1 \mid P_2) :: -{:}\mathbf{1}$ by rule $\multimap\mathsf{L}$

Case:$D = \mathsf{cut}\, D_1\, (y.\, D_2)$

$\Delta = (\Delta_1, \Delta_2)$, $P = (\nu y)(P_1 \mid P_2)$,
$\Gamma ; \Delta_1 \vdash D_1 \rightsquigarrow P_1 :: y{:}B$
$\Gamma ; \Delta_2, y{:}B \vdash D_2 \rightsquigarrow P_2 :: x{:}A$ by inversion
$\Gamma ; \Delta_2, y{:}B, x{:}\overline{A} \vdash P_2 :: -{:}\mathbf{1}$ by i.h. on D_2
$\Gamma ; \Delta_1, \Delta_2, x{:}\overline{A} \vdash (\nu y)(P_1 \mid P_2) :: -{:}\mathbf{1}$ by rule cut

Case:$D = \mathsf{cut}^!\, D_1\, (u.\, D_2)$

$P = (\nu u)(!u(z).P_1 \mid P_2)$,
$\Gamma ; \vdash D_1 \rightsquigarrow P_1 :: z{:}B$
$\Gamma, u{:}B ; \Delta \vdash D_2 \rightsquigarrow P_2 :: x{:}A$ by inversion
$\Gamma, u{:}B ; \Delta, x{:}\overline{A} \vdash P_2 :: -{:}\mathbf{1}$ by i.h. on D_2
$\Gamma ; \Delta, x{:}\overline{A} \vdash (\nu u)(!u(z).P_1 \mid P_2) :: -{:}\mathbf{1}$ by rule $\mathsf{cut}^!$

Case:$D = !\mathsf{R}$. This case is impossible, since A was assumed to be !-free.

□

References

Abramsky, S. (1993) Computational interpretations of linear logic. *Theoretical Computer Science* **111** (1-2) 3–57.

Andreoli, J.-M. (1992) Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation* **2** (3) 197–347.

Barber, A. (1997) Dual intuitionistic linear logic, *Technical Report LFCS-96-347*, University of Edinburgh.

Beffara, E. (2006) A concurrent model for linear logic. *Electronic Notes in Theoretical Computer Science* **155** 147–168.

Bellin, G. and Scott, P. (1994) On the π-calculus and linear logic. *Theoretical Computer Science* **135** 11–65.

Bonelli, E., Compagnoni, A. and Gunter, E. L. (2005) Correspondence assertions for process synchronization in concurrent communications. *Journal of Functional Programming* **15** (2) 219–247.

Boreale, M. (1998) On the expressiveness of internal mobility in name-passing calculi. *Theoretical Computer Science* **195** (2) 205–226.

Caires, L. (2007) Logical semantics of types for concurrency. In: Proceedings of the International Conference on Algebra and Coalgebra in Computer Science (CALCO'07). *Springer Lecture Notes in Computer Science* **4624** 16–35.

Caires, L., Pérez, J. A., Pfenning, F. and Toninho, B. (2012) Relational parametricity for polymorphic session types, *Technical Report CMU-CS-12-108*, Carnegie Mellon University.

Caires, L. and Pfenning, F. (2010) Session types as intuitionistic linear propositions. In: Proceedings of the 21st International Conference on Concurrency Theory, CONCUR'10. *Springer Lecture Notes in Computer Science* **6269** 222–236.

Carbone, M. and Debois, S. (2010) A graphical approach to progress for structured communication in web services. In: Proceedings of the 3rd Interaction and Concurrency Experience: Guaranteed Interaction (ICE'10). *Electronic Proceedings in Theoretical Computer Science* **38** 13–27.

Cervesato, I. and Pfenning, F. (2002) A linear logical framework. *Information and Computation* **179** (1) 19–75.

Chang, B.-Y. E., Chaudhuri, K. and Pfenning, F. (2003) A judgmental analysis of linear logic, *Technical Report CMU-CS-03-131R*, Carnegie Mellon University.

Dezani-Ciancaglini, M. and de' Liguoro, U. (2010) Sessions and session types: An overview. In: 6th International Workshop on Web Services and Formal Methods (WS-FM'09). *Springer-Verlag Lecture Notes in Computer Science* **6194** 1–28.

Dezani-Ciancaglini, M., de' Liguoro, U. and Yoshida, N. (2008) On progress for structured communications. In: Proceedings of the 3th Symposium Trustworthy Global Computing (TGC'07). *Springer Lecture Notes in Computer Science* **4912** 257–275.

Gay, S. and Hole, M. (2005) Subtyping for session types in the Pi calculus. *Acta Informatica* **42** (2-3) 191–225.

Girard, J.-Y. (1987) Linear logic. *Theoretical Computer Science* **50** 1–102.

Girard, J.-Y. and Lafont, Y. (1987) Linear logic and lazy computation. In: Theory and Practice of Software Development (TAPSOFT'87). *Springer Lecture Notes in Computer Science* **250** 52–66.

Giunti, M. and Vasconcelos, V. T. (2010) A linear account of session types in the Pi-calculus. In: Proceedings of the 21st International Conference on Concurrency Theorym (CONCUR'10). *Springer Lecture Notes in Computer Science* **6269** 432–446.

Honda, K. (1993) Types for dyadic interaction. In:4th International Conference on Concurrency Theory (CONCUR'93). *Springer Lecture Notes in Computer Science* **715** 509–523.

Honda, K. and Laurent, O. (2010) An exact correspondence between a typed pi-calculus and polarised proof-nets. *Theoretical Computer Science* **411** 2223–2238.

Honda, K., Vasconcelos, V. T. and Kubo, M. (1998) Language primitives and type discipline for structured communication-based programming. In: European Symposium on Programming (ESOP'98). *Springer Lecture Notes in Computer Science* **1381** 122–138.

Hyland, J. M. E. and Ong, C.-H. L. (1995) Pi-calculus, dialogue games and PCF. In: *WG2.8 Conference on Functional Programming Languages* 96–107.

Kobayashi, N. (1998) A partially deadlock-free typed process calculus. *ACM Transactions on Programming Languages and Systems* **20** (2) 436–482.

Kobayashi, N., Pierce, B. C. and Turner, D. N. (1996) Linearity and the pi-calculus. In: *23rd Symposium on Principles of Programming Languages (POPL'96)*, ACM 358–371.

Merro, M. and Sangiorgi, D. (2004) On asynchrony in name-passing calculi. *Mathematical Structures in Computer Science* **14** (5) 715–767.

Milner, R. (1992) Functions as processes. *Mathematical Structures in Computer Science* **2** (2) 119–141.

Pérez, J. A., Caires, L., Pfenning, F. and Toninho, B. (2012) Termination in session-based concurrency via linear logical relations. In: *22nd European Symposium on Programming (ESOP'12)* 539–558.

Pfenning, F., Caires, L. and Toninho, B. (2011) Proof-carrying code in a session-typed process calculus. *Certified Programs and Proofs (CPP'11)* 21–36.

Sangiorgi, D. (1996) Pi-calculus, internal mobility, and agent passing calculi. *Theoretical Computer Science* **167** (1-2) 235–274.

Sangiorgi, D. and Walker, D. (2001) *The π-Calculus: A Theory of Mobile Processes*, Cambridge University Press.

Toninho, B., Caires, L. and Pfenning, F. (2011) Dependent session types via intuitionistic linear type theory. In: *Principles and Practice of Declarative Programming (PPDP'11)*, 161–172.

Toninho, B., Caires, L. and Pfenning, F. (2012) Functions as session-typed processes. In: *15th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'12)* 346–360.

Wadler, P. (2012) Propositions as sessions. In: Proceedings of the *17th International Conference on Functional Programming, ICFP'12* 273–286.

Watkins, K., Cervesato, I., Pfenning, F. and Walker, D. (2004) Specifying properties of concurrent computations in CLF. In: Schürmann, C. (ed.) 4th International Workshop on Logical Frameworks and Meta-Languages (LFM'04). *Electronic Notes in Theoretical Computer Science* **199** 67–87.

Yoshida, N., Honda, K. and Berger, M. (2007) Linearity and bisimulation. *Journal of Logic and Algebraic Programming* **72** (2) 207–238.